SANDIA REPORT

SAND97–3190 • UC–705
Unlimited Release
Printed January 1998

Advanced 3D Electromagnetic and Particle-in-Cell Modeling on Structured/Unstructured Hybrid Grids

D. B. Seidel, M. F. Pasik, M. L. Kiefer, D. J. Riley, C. D. Turner

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Sandia National Laboratories

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A03
Microfiche copy: A01

SAND97-3190
Unlimited Release
Printed January 1998

Distribution
Category UC–705

Advanced 3D Electromagnetic and Particle-in-Cell Modeling on Structured/Unstructured Hybrid Grids

D. B. Seidel, M. F. Pasik, and M. L. Kiefer
Computational Electromagnetic and Plasma Physics Department

D. J. Riley and C. D. Turner
Radiation and Electromagnetic Analysis Department

Sandia National Laboratories
P. O. Box 5800
Albuquerque, NM 87185-1186

Abstract

New techniques have been recently developed that allow unstructured, free meshes to be embedded into standard 3-dimensional, rectilinear, finite-difference time-domain grids. The resulting hybrid-grid modeling capability allows the higher resolution and fidelity of modeling afforded by free meshes to be combined with the simplicity and efficiency of rectilinear techniques. Integration of these new methods into the full-featured, general-purpose QUICKSILVER electromagnetic, Particle-In-Cell (PIC) code provides new modeling capability for a wide variety of electromagnetic and plasma physics problems. To completely exploit the integration of this technology into QUICKSILVER for applications requiring the self-consistent treatment of charged particles, this project has extended existing PIC methods for operation on these hybrid unstructured/rectilinear meshes. Several technical issues had to be addressed in order to accomplish this goal, including the location of particles on the unstructured mesh after transport, the allocation of each particle's current and charge to the unstructured mesh, adequate conservation of charge, and the proper handling of particles in the transition region between structured and unstructured portions of the hybrid grid.

Acknowledgments

The authors wish to acknowledge the contributions and support of several persons. Discussions with Scott Brandon at Lawrence Livermore National Laboratory and John Ambrosiano at Los Alamos National Laboratory (formerly with Lawrence Livermore National Laboratory), who had both done initial ground-breaking work on applying PIC techniques to unstructured grids, were quite useful and helped guide our route to one less rocky than might otherwise have been the case. Jeff Quintenz has been instrumental in fostering the development of PIC simulation tools at Sandia and that support has continued through the course of this project. Finally, we would like to acknowledge Sandia's LDRD program which provided the funding for this project and at least part of the seed for the next generation of Sandia PIC tools.

Contents

Figures

Introduction

A significant impediment to the use of time-dependent, full-wave electromagnetic (EM) codes for the simulation of moderately complex devices is the limited resolution that can be achieved affordably with currently available computers. This is particularly true in three dimensions, where even small improvements in resolution quickly consume available processor and memory resources for conventional structured-grid, rectilinear finite-difference algorithms. When such codes are extended to self-consistently treat the motion of charged particles using Particle-in-Cell (PIC) methods,[1] the conventional rectilinear-grid approach, because of its "stair-stepped" approximation of complex surfaces, limits our ability to resolve particle motion near such surfaces accurately.

In order to circumvent this problem, previous work has focused on the development of entirely new algorithms based upon both structured non-orthogonal grids[2,3] and unstructured grids.[4,5] However, these approaches have shortcomings, most notably that the considerable investment in existing rectilinear-grid codes is lost and that, for most applications, the added overhead of unstructured grids is wasted on all but a small fraction of the simulation volume. However, new techniques have been recently developed that allow unstructured, free meshes to be embedded into standard 3-dimensional, rectilinear, finite-difference time-domain grids.[6] The resulting hybrid-grid modeling capability allows the higher resolution and fidelity of modeling afforded by free meshes to be combined with the simplicity and efficiency of rectilinear techniques.

This report details the issues and methods involved in extending standard PIC techniques to a hybrid grid. This has been accomplished by modifying the rectilinear structured grid PIC code QUICKSILVER[7,8] to include the unstructured-grid EM solver VOLMAX[6,9,10] and then generalizing and extending QUICKSILVER's particle handling methods to the unstructured grid. This extension presented many challenging problems that needed to be solved before a satisfactory implementation could be achieved. These problems included the space-charge-limited emission of particles, locating particles on the unstructured grid after transport, allocating each particle's current and charge to the mesh, adequately conserving charge, and proper particle treatment in the transition regions between structured and unstructured portions of the hybrid grid.

Background

The work described in this report is built upon the two Sandia-developed simulation codes: QUICKSILVER and VOLMAX. A brief summary of the capabilities and features of those two codes follows.

QUICKSILVER: A Time-Dependent, Finite-Difference EM PIC Code

QUICKSILVER is a three-dimensional (3D), time-dependent, EM PIC code whose field-solving algorithm is based upon a finite-difference formulation of Maxwell's equations on a

multiple-block, rectilinear structured grid. A structured grid is one in which every grid element is associated with its nearest neighboring grid elements though a logical mapping to a three-dimensional cubical lattice. A rectilinear structured grid also has the property that its elements are arranged conformal to a standard orthogonal coordinate system;, e.g., QUICKSILVER's grid is conformal to Cartesian, cylindrical, or spherical coordinates. Other codes (see, for example, Refs. 2 and 3) are implemented on more general structured grids that are neither coordinate-system-conformal nor orthogonal. By multiple block, we mean that the grid is composed of logically connected blocks, each of which is a coordinate-system-conformal region of space with its own local grid. Each block has a small region of overlap (two cells) with its neighboring blocks which allows the fields in each block to be advanced in time independently; the fields at the outer edges of a block are then supplied as a boundary condition from the overlap region of the appropriate neighboring block.

QUICKSILVER is actually a suite of codes; in addition to the main simulation code there are several support codes. The problem geometry is generated using various preprocessors and the simulation results are examined with one or more postprocessors. The original MERCURY command-driven preprocessor assists the user in defining the mesh, boundary conditions, and other input parameters. Recently, a set of widget-based tools has been developed to further simplify the process of mesh generation. These widget-based tools, built upon the IDL data analysis and visualization tool[†] have been incorporated in PFIDL[11], QUICKSILVER's primary simulation data postprocessor. In its role as postprocessor, PFIDL provides the capability to manipulate and examine 3D scalar and vector field data as well as 6D particle phasespace data. Additionally, PFIDL can be used to examine and manipulate time histories of various simulation quantities. AVS[‡] is used for the visualization and/or animation of field and particle distributions as well as the 3D model geometry. These pre- and postprocessing tools are available on a wide variety of platforms. The potentially vast amount of simulation data is shared between the simulation code and the postprocessors via the Portable File Format (PFF)[12], a portable, compact, machine-independent binary file format developed expressly for the QUICKSILVER suite but widely used for many other applications.

Generating input data for three-dimensional simulations can be difficult, time-consuming, and error-prone. MERCURY is a command-driven preprocessor that is used in defining the finite-difference grid, the problem geometry, the boundary conditions, and other input parameters. MERCURY allows free-format input and provides menus for guiding simulation setup and on-line help. It processes all input for a QUICKSILVER simulation and checks for errors and inconsistencies. QUICKSILVER uses a nonuniform, multiple-block, rectilinear grid with staggered grids. The MERCURY grid generator provides straightforward tools to facilitate the generation of these multi-block, nonuniform grids, automatically ensuring that the grid is both continuous and smoothly varying. Cartesian, cylindrical, and spherical coordinate system multiple-block grids can be generated. Conducting and dielectric volumes are easily generated with MERCURY by combining (sequentially adding or removing) objects selected from a provided set of simple solid-object primitives. MERCURY then fits the resulting compound volume description to the simulation's underlying finite-difference grid.

[†]IDL is a product of Research Systems, Inc., 2995 Wilderness Place, Suite 203, Boulder, Colorado 80301.

[‡]AVS is a product of Advanced Visual Systems, 300 Fifth Ave., Waltham, MA 02154.

MERCURY also computes the memory requirements for arrays in QUICKSILVER so that only the minimum memory required for a simulation is used.

In addition to MERCURY's capabilities, widget-based tools have been added to PFIDL to ease some of the more difficult facets of simulation setup. For example, since often a description of the problem geometry is available from solid modeling or CAD tools, PFIDL currently has a tool for editing DXF[†] or ACIS[‡] files and converting them to a format compatible with MERCURY.

QUICKSILVER, the member of the suite for performing 3D physics simulations, can be divided into two distinct parts, the field solver and the particle handler. The QUICKSILVER field solver utilizes explicit[13,14] and implicit[15] finite-difference, leap-frog algorithms. A single cell of a QUICKSILVER grid is depicted in Figure 1, which shows the staggered spatial locations of the electric and magnetic field components in the differencing algorithm. The *I*, *J*, and *K* subscripts on the field components indicate the corresponding coordinate direction in one of the three supported coordinate systems. Multiple lossy, non-dispersive dielectrics are allowed for regions without particles. Available boundary conditions include conductors, inlet and outlet boundaries, mirror symmetry, and periodic symmetry. Currently, inlet wave boundaries can be driven either with multiple, independent TEM modes or a 1D, multi-line Telegraphers' model. In both cases, outgoing waves are treated with a 1st-order Mur-like[16] radiation-absorbing boundary condition.[17] The code also supports other outlet boundary conditions, i.e., 2nd-order dispersive[18] and the Perfectly Matched Layer (PML).[19] QUICKSILVER also has models for embedded current source excitation and surface impedance boundary conditions.

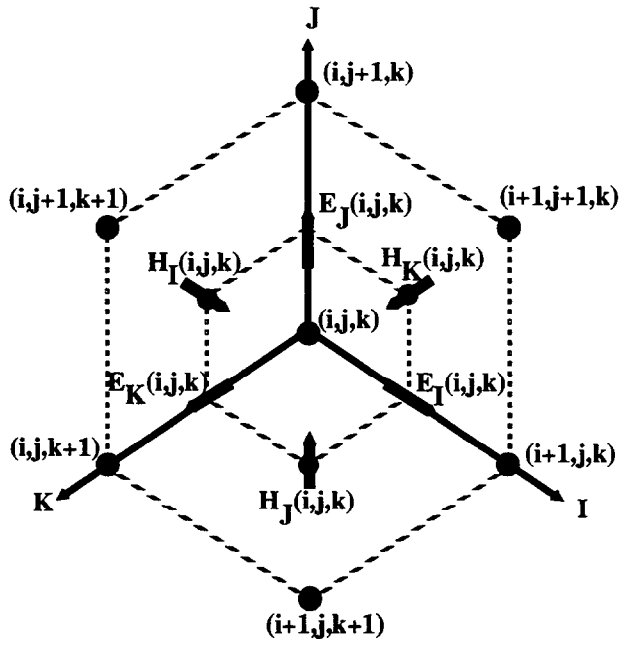

Figure 1. Spatial location of electric and magnetic field components in a single cell of the QUICKSILVER grid.

The second major portion of the QUICKSILVER code is its particle handler, whose job is to advance particle positions with 3D, fully-relativistic kinematics and to subsequently allocate each particle's contribution to the current back to the finite-difference grid for use by the field solver. QUICKSILVER's particle handler allows multiple particle species with particle creation via preloading, beam injection and space-charge-limited (SCL) field emission. It supports the same boundary conditions and coordinate systems as the field solver. Currently the code uses

[†]DXF is a registered trademark of Autodesk, Inc., 111 McInnis Parkway, San Rafael, California 94903.

[‡]ACIS is a registered trademark of Spatial Technology, Inc., 2425 55th St., Bldg. A, Boulder, Colorado 80301.

a current/charge density allocation algorithm that locally conserves charge exactly.

The QUICKSILVER code has a wide variety of diagnostics available to the user which can be divided into two basic types: snapshots and time histories. Snapshot diagnostics provide detailed spatial information about some simulation quantity at specified instants of time (or averaged over specified intervals of time). On the other hand, time histories provide, as a function of time, a simulation quantity at a fixed spatial location or integrated over some spatial region of the simulation. QUICKSILVER can provide snapshots of both vector and scalar field quantities as well as snapshots of simulation particles in 6D phasespace (x, y, z, p_x, p_y, p_z), or a subset of that phasespace. Time histories can be requested for ρ or any component of **E**, **B**, or **J** at any spatial location in the simulation. In addition, line, surface, and volume integrals are available, each over one or more coordinate-conformal subpaths, subareas, or subvolumes, respectively. Time histories are also available for several particle-related items, including count, energy, or charge of surviving, created, or killed particles, by species. To examine simulation charge conservation, maximum and RMS values of the error in charge conservation ($\nabla \cdot \mathbf{D} - \rho$) are also available as time histories.

VOLMAX: An Unstructured Grid, Time-Dependent, Finite-Volume EM Simulation Code

VOLMAX is a three-dimensional transient volumetric Maxwell equation solver that operates on standard rectilinear finite-difference time-domain (FDTD) grids, non-orthogonal, unstructured finite-volume time-domain (FVTD) grids, or a combination of both types (hybrid grids). The algorithm is fully explicit. Systems are typically simulated by embedding multiple unstructured regions into a simple rectilinear FDTD mesh. Boundary conditions are supplied to the system on the exterior FDTD mesh. A wide variety of boundary conditions are available on the structured grid, comparable to those described for QUICKSILVER in the previous section. The grid types are fully connected at the mesh interfaces without the need for complex spatial interpolation. The approach permits detailed modeling of complex geometry while mitigating the large cell count typical of non-orthogonal cells such as tetrahedral elements. To further improve efficiency, the unstructured region carries a separate time step that sub-cycles relative to the time-step used in the FDTD mesh. A cross section of the interface between the FVTD and FDTD grids is shown in Figure 2. The "wrapper layer" is a hexahedral region that encloses the unstructured grid and provides nodal connectivity to the surrounding FDTD mesh. The wrapper is constructed automatically based on the unstructured-grid topology. The unstructured region may consist of a single rectangular block, or be of a multiple, block-on-block form.

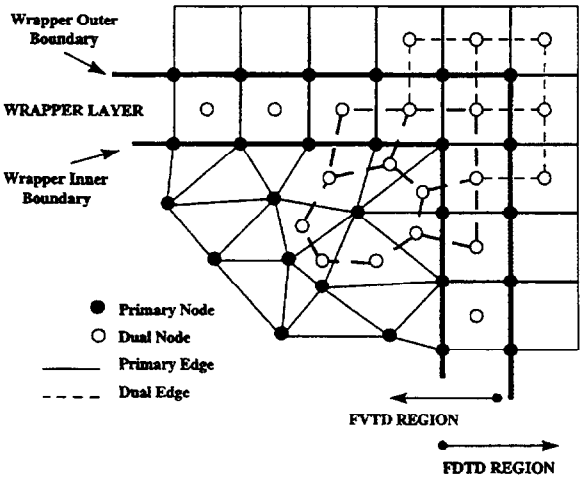

Figure 2. The VOLMAX hybrid grid interface.

As shown in Fig. 2, VOLMAX is based on a staggered grid formulation. Primary and dual grids are used. The dual grid is constructed automatically based on the topology of the user-specified primary grid. Note that the wrapper layer consists of rectangular cells for its primary grid, but the dual cells on the wrapper inner boundary are generally non-orthogonal. As a consequence, the wrapper layer is common to both the FVTD and FDTD grids. For the case that the unstructured-grid consists of uniform rectangular elements, the algorithm is equivalent to the FDTD algorithm used by QUICKSILVER (even though the cells are referenced in an unstructured manner) and is second-order accurate both in space and time.

For the unstructured portion of the mesh, VOLMAX must maintain a database that describes the various properties of the elements of the mesh as well as the relationships between mesh elements that describe the mesh's connectivity. This is accomplished by maintaining a list for each primary cell of the primary faces that enclose that cell. Then, for each primary face, it maintains a list of the primary edges that define that face. Finally, for each primary edge, there is a list of the two primary nodes defining the edge. Similar lists describing the relationships of the dual mesh elements are maintained. To relate the primary and dual grids, mappings between their corresponding elements are also maintained. For example, for each primary cell there is a corresponding dual node; for each primary face there is a corresponding dual edge, etc. Note that this means, for example, that to find the primary nodes of a primary cell, you must successively traverse the list of faces for that cell, then the list of edges for each face, and finally, the list of nodes for each edge. The VOLMAX database also contains property information, such as cell volumes, face areas and normals, node location, etc.

The field advancement scheme for the VOLMAX hybrid mesh is the following. The electric fields in the FDTD region are initially advanced based on time step, Δt. On the outer boundary of the wrapper, the tangential electric fields are second-order time interpolated to provide a Dirichlet boundary condition for the FVTD region. The electric and magnetic fields in the FVTD region are advanced an integral number (N_U) of sub-time iterations relative to Δt. At the completion of the sub-cycling, the tangential electric fields on the inner boundary of the wrapper are used to provide a Dirichlet boundary condition to complete the magnetic-field advancement in the FDTD region. An alternative scheme could map the magnetic fields in the wrapper layer into the respective FDTD locations after the FDTD magnetic fields are advanced in time.

VOLMAX is currently integrated to the commercial CAD package SDRC I-DEAS[†]. Solid model design, mesh generation, and post-processing are all accomplished through the I-DEAS interface. Electromagnetic properties, such as voltage sources, local boundary conditions, current observers, input and output ports, slots, wires, etc., are implemented by assigning nodal attributes. The file containing the primary grid generated by I-DEAS is input into the VOLMAX preprocessor, PREVOL, which builds the wrapper layer, completes the connectivity for the primary grid, and constructs the dual grid. Grid construction by PREVOL is accomplished at the rate of 50,000 to 100,000 cells/minute on a single, high-end processor.[20] Construction time scales linearly with cell count.

[†]I-DEAS is a product of Structural Dynamics Research Corporation, Milford, Ohio.

Outline of Report

The development required to extend PIC methods to hybrid grid EM solutions can be divided into three distinct categories. First, the new issues associated with pushing particles on unstructured grids must be addressed. This includes determining a particle's new location on the mesh after it has been advanced in space, interpolation of mesh quantities to and from a particle's position, and dealing with new complications regarding the non-conservation of charge. Second, these additions and modifications must be properly synchronized in time. A third category involves particle treatment and significant new complications in field continuity at the hybrid grid interface. Each of these areas will be elaborated in detail in subsequent sections of the report. Additionally, there will be sections describing algorithm testing and performance as well as a short discussion of current status and future direction.

Unstructured Grid Particle Handling

There are several specific components of any algorithm that handles the motion of PIC particles and their interaction with the fields on the underlying computational mesh. In this section, we will address the issues and complications associated with extending each of these components to properly treat particles when the underlying grid is unstructured. To the extent possible, we will defer until later in the report discussion of the integration of these components with each other as well as their integration with their structured-grid counterparts.

Particle Location on Unstructured Mesh

In a PIC code, it is important to be able to determine a particle's location relative to the grid so that the particle's effect upon field quantities defined on the grid and the effect of the fields on the particle can be ascertained. In typical PIC codes on rectilinear, structured grids, determining a particle's location (what cell is it now in) after its position has been advanced in time is a straightforward, almost trivial exercise — typically a particle is constrained to move no more than one cell by Courant stability requirements[21] of the field solution, so we simply need to check if it crossed the upper or lower cell boundary in one or more of the three coordinate directions. If it has left the cell, we can easily determine into which cell it has moved because of the underlying implied grid connectivity due to the separability of the grid in the three coordinate directions.

Contrast this to the same question on an unstructured grid. As with the rectilinear grid, we can determine if a particle has left a cell by checking if it has crossed any one of the faces of the cell. However, since the faces are not in general conformal to the coordinate system, testing for this condition is much more complicated and time-consuming than the simple comparison of one component of the particle's spatial position with a cell grid value. In addition, faces with more than three nodes are in general *not* planar, so even the concept of upon which side of a face a particle resides is not clearly defined without a convention for subdividing such a face into a set of planar facets.

The prescription for determining whether or not a particle is beyond one of a cell's faces is straightforward for planar faces. If a vector $\mathbf{V}_P$ is constructed from any point in the plane of the face (e.g., one of the face's nodes) to the particle location, and the dot-product of this vector with the face's outward normal is positive, then the particle is beyond that face. This is shown in 2D in Figure 3, which graphically shows the positive dot product of the test

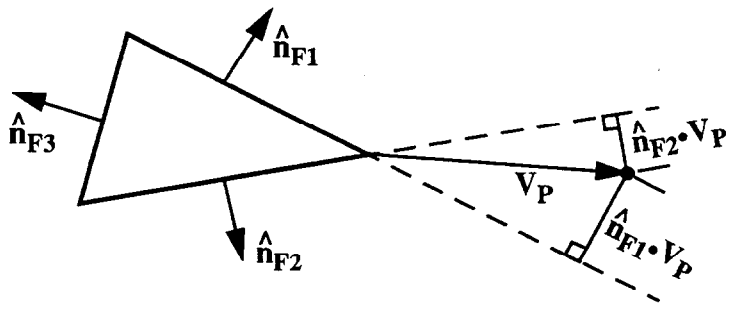

Figure 3. 2D example of using dot products to determine upon which side of a face a particle is located.

vector $\mathbf{V}_P$ with the normals of two of the three faces. Note that the dot product with the third face's outward normal is clearly negative. In the special case that a particle is located precisely in the face between two cells, there is a potential for floating point roundoff to cause problems. The worst scenario is that due to roundoff the algorithm would determine the particle to be slightly outside of *both* cells, and any algorithm to find the cell containing the particle would certainly fail. One the other hand, if the particle were so close to a face that both cells sharing the face would claim the particle, it would not really matter to which cell the particle was assigned. As will be seen in the next section, the particle's weighting factors to nearest nodes would be essentially the same since the only non-vanishing weights would be to the nodes of the shared face. Consequently, the actual algorithm to determine if a particle is outside a cell compares the face dot-products to a small positive value (small compared to the average length of grid edges) rather than to zero.

Once it has been determined that a particle is beyond one or more of the original cell's faces, we still must find in which cell the particle now resides. Since cells in an unstructured grid aren't arranged in a lattice, we require a mapping from each cell's face to the neighboring cell that shares that face. (Such a mapping is available in the VOLMAX code). However, it is possible that the particle is not in the neighboring cell, so we must go through the same process of comparing particle position to the new cell's faces to find if the particle is in that cell, or if not, another cell in which to look. This process is continued until we find a cell containing the particle. We will use the obvious strategy to optimize such a search of choosing the next cell in the search based upon which dot product was most positive.

This is best illustrated by example. Figure 4 shows a simple 2D example of a portion of an unstructured grid. Also shown is a particle that has moved from $\mathbf{x}_i$ in cell 1 to $\mathbf{x}_f$ in cell 4 along a straight path. Note that its final position is beyond both the upper and right face of cell 1; consequently our search algorithm could take us first to cell 2 or to cell 5. In this case, since the dot-product is most positive for face shared with cell 5, it would be selected. It is easy to see that in either case, successive application would eventually lead us to cell 4.

So far we have not considered the effects of model structure upon particle location. Consider the 2D example shown in Figure 5. This is identical to the first example *except* that cell 5 no longer exists, and cells 1 and 4 now each have faces that are not shared with other cells but are instead at a simulation boundary, typically a perfect conductor. A particle

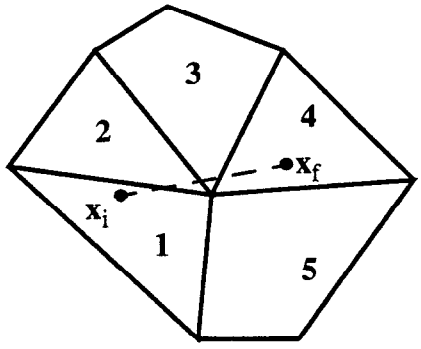

Figure 4. Simple 2D example of unstructured grid particle location.

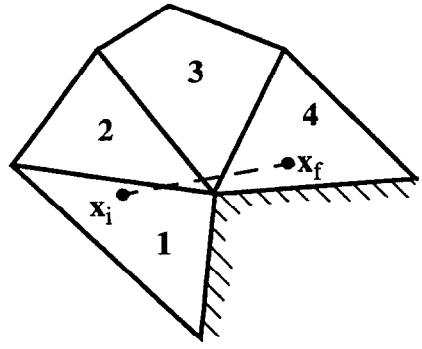

Figure 5. Simple example of particle location in the presence of model structure.

encountering such a face would need to be removed from the simulation after allocating the current associated with its motion from its initial position to the point where its path intersects the boundary face. It should be noted that this same situation could exist in the example of Figure 4 if cell 5 were tagged with a special material attribute indicating it was a perfect conductor. If the dot-product algorithm is applied in this case, we would first attempt to cross the boundary face on the right side of cell 1. However, a critical question becomes whether or not the particle's path actually intersects the face. If it does, we simply need to find the intersection point and remove the particle from the simulation. If, as shown in the example, it doesn't intersect the face, we need to look for an alternate path from cell 1 (in this case through cells 2 and 3) to find the particle's final location. An important issue is how to efficiently determine if a particle's path intersects a cell face. The method we eventually settled upon first determines the intersection point between the path and the plane containing the face. Note that if the path in fact intersects the actual face, this location is needed anyway. We next determine whether or not this new point lies within the original cell using the dot-product algorithm, which for the reason described earlier, will determine that a particle located within numerical roundoff of one of a cell's faces is within that cell. If the point is found to be within the cell, the path intersects the face — if not, we must look for an alternate path through another face of the original cell.

We presently have an implementation of this algorithm that is being successfully used to locate particles on unstructured meshes. It is presently limited to meshes whose faces are planar. It should be noted that several mesh-related quantities are needed for the computations required:

- a list of faces for each cell,
- a list containing the neighboring cell, if it exists, for each face; otherwise an indication that the face is a boundary face,
- a normal for each face and a means to determine whether it is inward or outward relative to the cells sharing the face,
- a list of nodes and their locations for each face.

The original VOLMAX field solver provided most of these quantities. The major exception was that there was no way to determine the orientation of a face's normal relative to cells containing the face. To accommodate this new requirement, minor modifications were required in VOLMAX's mesh database. Basically, the sign of the face number on the list of a cell's faces is now used as a multiplier of the face normal to insure that it is outward directed.

Weighting Factors for Particle/Mesh Interaction

PIC codes require weighting factors to account for interaction between simulation particles and the simulation fields located on the finite-difference grid. These factors are used to interpolate field values from the grid to the particle's spatial location, as well as to allocate the current and charge associated with a particle and its motion to the grid-based current and charge density fields. In typical PIC codes on rectilinear, structured grids, computing the weighting factors is straightforward; they are simply the product of linear weights in the three coordinate directions. For example, in 1D, for a particle at x_P located between two grid points x_1 and x_2, the linear weights are simply:

$$W_1 = \frac{x_2 - x_P}{x_2 - x_1} \text{ and } W_2 = \frac{x_P - x_1}{x_2 - x_1}. \quad (1)$$

In three dimensions, there are eight weights associated with the eight nodes (corners) of the right-hexahedral cell, corresponding to the eight permutations of the product of the 1D weights in the three coordinate directions. These weights have several important properties. First, their sum is always one. Second, as the particle approaches any one of the eight grid nodes, the weight associated with that node approaches one. As the particle approaches any face (or edge) of the cell, only weights associated with the nodes on that face (or edge) are non-zero. In a qualitative sense, there is a sensible correlation between the magnitude of a node's weight and the proximity of that node to the particle. Finally, their computation is reasonably simple and efficient.

On a fully general unstructured grid, the situation is much more complex. For the most general types of unstructured-grid cells, developing a weighting scheme that possesses the attributes described above is quite difficult, and any implementation would be extremely computationally intensive. However, if we restrict ourselves to a limited subset of cell types, the task is much more tractable. We have chosen to implement weighting schemes for three basic cell types: conformal linear right hexahedra, extruded linear triangular prisms (wedges), and linear tetrahedra. This subset still allows efficient modeling of very complex geometric structures and has the benefit that the faces of these cell types are planar. It should be noted, however, that VOLMAX's dual grid, which is generated automatically as a result of topology of the primary grid, will in almost all cases be composed of a bewildering array of cell-types, most of which are not among the three types to which we have limited ourselves. Consequently, we have restricted ourselves to weighting to and from primary nodes. Thus, any field quantities that are involved in such weighting need to be mapped to primary nodes. This means that current density and charge will be located at primary nodes. In addition, magnetic fields are located at the *dual* nodes in VOLMAX's field solver. Since we need to interpolate magnetic field to a particle's spatial location in order to calculate the forces on that particle, we will need to first obtain an "average" magnetic field from the dual node values at the primary nodes.

Before describing the weighting algorithm for each of these three cell types, it is useful to discuss some issues that pertain to weight computations in general. For the computation of weighting factors as well as for their subsequent application, it is important that the nodes associated with a cell are known, and even more, that each node's relationship to the cell is known. Unfortunately, since this is irrelevant to the EM field solution, the original version of VOLMAX did not directly store this information. Instead, it maintained multiple lists relating cells to faces, faces to edges, and edges to nodes as described earlier. Unfortunately, traversing each list in succession generally finds every edge twice and finds every node a multiple, indeterminate number of times. In addition, there is no topological information that relates the nodes to the original cell. Since all this information is typically required for every simulation particle one to two times per simulation timestep, the computational cost is staggering and it is quickly apparent that an addition was required to the VOLMAX mesh database. This addition was simply a new list that provided the nodes associated with a cell for each primary cell in the simulation. This list is constructed by traversing the multiple lists as described above and

removing the multiple node instances. At the same time, the nodes are ordered by conventions distinct to each cell type, so that their topological relationship to the cell can be inferred. In addition, a CellInfo array was added that contains tags that indicate various properties of each cell. A "cell-type" tag was then associated with each cell so the weighting (and location) algorithms can easily utilize cell-type-specific methods. It should be noted that the code required to construct this list is extremely complex and requires significant computation. Without constructing this cell-to-node mapping *a priori*, the cost of determining it every time a particle is processed would be prohibitive.

Weighting for Conformal Linear Right Hexahedra

Conformal right hexahedra are exactly the cell type used in a standard rectilinear grid PIC code. These cells are needed for two reasons: first, the "wrapper" cells that connect the structured and unstructured regions of the grid (cf. Figure 2) are by necessity of this type, and second, it is useful for the purposes of testing and validation to construct an unstructured equivalent of a structured rectilinear grid. Since this is just a standard rectilinear cell we can use the linear weighting scheme described in (1), the only complication being that we need to know the location of the upper and lower corners ($\mathbf{x}_u$ and $\mathbf{x}_l$) of the cell. This is accomplished be ordering the eight nodes based upon their relative position in the three coordinate directions, i.e., (x_l, y_l, z_l), (x_u, y_l, z_l), (x_l, y_u, z_l), ..., (x_u, y_u, z_u). Consequently, the coordinate limits of the cell are defined by the spatial locations of its first and eighth nodes.

Weighting for Extruded Linear Triangular Prisms

Extruded triangular prisms, or wedges, are useful for modeling complex structures that have translational or rotational symmetry. Weights for these cells are constructed as a product of two weights, a 1D weight in the direction of the extrusion and a 2D transverse weight over the triangular cross-section of the prism. The 1D weight is again the simple linear weight described in (1), the complication being that we need to know the coordinate direction (x, y, or z) of the extrusion and the spatial range of the cell in that coordinate. Presently, the extrusion orientation is contained in the cell's "cell-type" tag, i.e., there are three prism types: X-prisms, Y-prisms, and Z-prisms. However, it would be relatively easy to extend this to support more general non-conformal extrusion orientations. The range of the cell in the extrusion coordinate is determined implicitly by the cell ordering — the first three nodes are located at the "low" end of the extrusion, the remaining three nodes at the "high" end. In addition, the nodes at both ends of the extrusion are arranged in the same order relative to the 2D triangular cross-section.

Computation of the transverse weighting is based upon an area-weighting concept similar to that which can be used to describe the standard rectilinear method. Figure 6 shows a particle's location ($\mathbf{x}_p$) in the transverse cross-section of the prism. Three sub-triangles are constructed by connecting the particle location to each of the three nodes. The weight of each node is then taken to be the ratio of the area of the sub-triangle opposite that node to the area of the entire triangle. For example, in Figure 6, the transverse weight associated with nodes 1, 2, and 3 would be A_1/A, A_2/A, and A_3/A, respectively, where A_1, A_2, and A_3 are the areas of the three sub-triangles and A is the area of the entire triangle. Note that these would also be

the transverse weights of nodes 4, 5, and 6 (the nodes extruded from 1, 2, and 3), respectively. Also note that these weights possess all the characteristics previously described.

To compute the areas of the sub-triangles, we can use the face dot-products computed previously (cf. Figure 3) in the process of particle location. The height of the particle above the primary edge that forms the base of a sub-triangle is the negative of the dot-product for the face extruded from that edge. With this height, the sub-triangle's area is easily computed, e.g.,

$$A_1 = -\frac{1}{2}\frac{A_F}{L}(\hat{\mathbf{n}}_F \cdot \mathbf{V}_P),$$

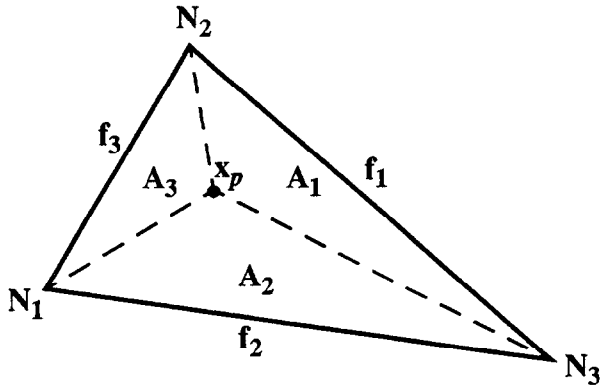

Figure 6. Transverse cross-section of an extruded triangular prism showing the transverse weighting scheme.

where L is the extrusion length, A_F is the area of the extruded face opposite node 1, and the dot-product is the face dot product for that same face. Since A can be expressed as the ratio of the cell volume to the extrusion length L, the transverse weighting factor becomes

$$W_{trans} = -\frac{1}{2}\frac{A_F}{V_C}(\hat{\mathbf{n}}_F \cdot \mathbf{V}_P),$$

where V_C is the volume of the extruded cell. VOLMAX already stores both the area of all faces and the volume of all cells. Also note that we have assumed an implicit relationship between nodes and the extruded faces. This is in fact accomplished by reordering the list of faces for each cell of this type so that the three extruded faces are first ordered to correspond with the order of the nodes opposite those faces on the list of nodes for the cell. The remaining lower and upper triangular faces are ordered as faces 4 and 5, respectively.

Weighting for Linear Tetrahedra

Tetrahedral cells are the most common type of non-orthogonal cell used in VOLMAX and provide the most modeling flexibility. The weighting scheme employed for tetrahedra is the 3D extension of the method used for the transverse weighting in wedge cells described in the last section. In this case, sub-triangles become sub-tetrahedra and the weights become the ratios of the volumes of these sub-tetrahedra to the volume of the entire tetrahedral cell. Figure 7 shows a particle's location ($\mathbf{x}_p$) in a tetrahedral cell. A_1 is the area of face 1. The four nodes associated with the cell

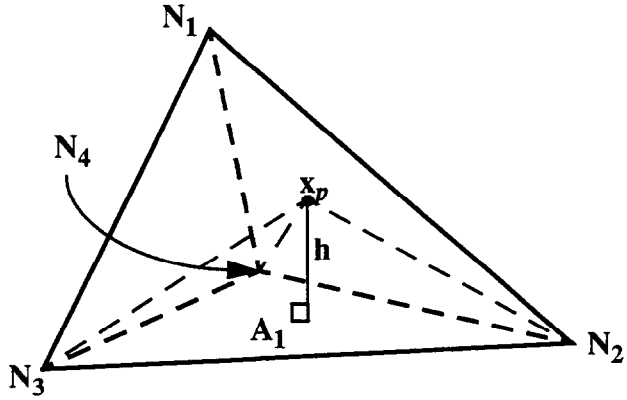

Figure 7. Depiction of the weighting algorithm for a linear tetrahedron.

have been ordered so that each node is opposite to the corresponding face on the list of faces for the cell. A sub-tetrahedra (whose base is opposite node 1) and its height h are also shown. Note that all the criteria for weight factors are satisfied by this scheme. As in the case of the wedge cell, we can make use of previously computed face dot-products to compute the weights. Knowing that the volume of a tetrahedron is one-third the product of the area of its base and its height, we obtain:

$$W_1 = -\frac{1}{3}\frac{A_F}{V_C}(\hat{\mathbf{n}}_F \cdot \mathbf{V}_P),$$

where V_C is the volume of the tetrahedral cell, A_F is the area of the face opposite node 1, and the dot-product term is the face dot-product for that same face.

Current and Charge Allocation

After the particles in a PIC simulation have been advanced in time, their motion, which reflects a current, needs to be apportioned to the simulation grid. This provides the means by which the EM fields are affected by the motion of the free charge in the system. The choice of algorithm for this allocation is always a trade-off between the factors of accuracy, numerical discretization noise, and computational efficiency. For example, in a standard rectilinear PIC code, an algorithm might be efficient and conserve charge but introduces unwanted numerical noise. On the other hand, a different approach may be relatively noise-free, but introduces significant errors in charge conservation. In general, both of these problems are increasingly mitigated as the computational complexity and load of the allocation scheme are increased. For example, the QUICKSILVER allocation scheme opts to conserve charge and be reasonably efficient at the cost of significant numerical noise. This undesirable result is then mitigated through modifications to the EM field algorithm that provide low-pass filtering to reduce the noise.

Although charge conservation is certainly desirable, implementing a charge-conserving allocation algorithm on a general unstructured grid is significantly more complicated and expensive; consequently, we have decided to use a less expensive, less-noisy algorithm. This pushes the issue of charge conservation to another part of the code. A detailed discussion of this issue is deferred to the next section of the report.

As described in the previous section, other choices have restricted us to allocate both current and charge to primary nodes. However, we still have some freedom on exactly how this is done. For example, consider the case of a particle moving through the unstructured grid shown in Figure 4. Allocation of charge is reasonably straightforward since charge and particle position are co-located in time. The obvious choice (and that made for most rectilinear algorithms, including QUICKSILVER) is to allocate the entire charge of a particle to the cell associated with its final position, using the nodal weights appropriate for that cell's type. In the example of Figure 4, this means the charge would be entirely allocated to the nodes of cell 4. Allocation of current is more difficult. In our example, the particle moves through cells 1, 2, 3, and 4. Since current is located in time one-half timestep away from particle position, one possibility (used in many rectilinear codes) would be to allocate current to the cell that

contains the midpoint of the path between the particle's initial and final positions. Note that this in general requires the computation on an additional set of weighting coefficients for this intermediate cell, and also requires that the particle location algorithm be used to find two, rather than one, particle positions for each particle at each timestep. An even more expensive but probably less noisy method would be to divide the particle's path into subpaths that are each confined to only one cell and then allocate some fraction of the particle's motion to each path. In our example, current would be allocated separately to all the nodes in cells 1–4. This is clearly much more expensive since you would now need to find the intersection of a particle's path with every face it crossed rather than only boundary faces. In addition, weighting factors would need to be computed for every cell along the path. Any sensible means of apportioning the current to each cell would probably require computation of each of the subpath lengths as well.

Because particles are constrained by the field solver's Courant limit to move at most some fraction of the average edge length, multiple face crossings, as in the case of our example, are limited to situations where the particle's path cuts through corners of cells very near a vertex that is shared by the initial and final cells. This means that contributions to nodes of cells other than the initial and final cells would be small by the very nature of the weight factors. Consequently, we have chosen to initially implement yet another variation, i.e., we will allocate one-half of the particle's current in the initial cell and one-half in the final cell. In our example, we would allocate to the nodes of cells 1 and 4. Note that this does not require any additional location or weight computation. Also note that in the case that a particle does not actually leave its starting cell, all these schemes are equivalent. In the case that its path falls in only two adjacent cells, it is less noisy (has a larger stencil) than the midpoint method, and is the same as the subpath method except for the relative apportionment of current to the two cells. In the case that more than two cells are traversed, the subpath method is probably the least noisy.

Next a brief discussion of the mechanics of allocation is in order. As particles move through the grid, their charge is accumulated to the appropriate primary nodes according to the weighting factors. After all particles have been advanced for a given timestep, the charge accumulated at each primary node represents the total charge at that node, or alternately, the total charge within the dual cell containing the primary node, and is given by

$$Q_N = \sum_P q_P W_{N,P},$$

where q_P is each particle's charge and $W_{N,P}$ is the weighting factor for node N and particle P. In contrast to the structured-grid algorithm, we choose to save the charge, rather than the charge density, at each node. However, to get the charge density at the node, we simply divide the allocated charge by the volume (V_D) of the dual cell corresponding to the node.

Current density is somewhat more complicated. Since the current density is simply the product of the charge density and the particle velocity, we can express the current density associated with a single particle as

$$\mathbf{J} = \rho \mathbf{v} = [q_P/V]\ [(\mathbf{x}_f - \mathbf{x}_i)/\Delta t],$$

where $\mathbf{x}_i$ and $\mathbf{x}_f$ are its initial and final position, Δt is the timestep, and V is the volume

occupied by the particle. Consequently, as the particles are advanced, we accumulate at each node N to which the particle contributes the vector quantity

$$\mathbf{T}_N = \sum_P q_P W_{N,P}(\mathbf{x}_f - \mathbf{x}_i).$$

After all particles have been advanced for a given timestep, we then convert $\mathbf{T}$ at each node to current density as follows:

$$\mathbf{J}_N = \mathbf{T}_N/(2V_D\Delta t), \tag{2}$$

where V_D is the volume of the dual cell containing node N and the factor of two accounts for the fact that we have allocated one-half of each particle's motion to the nodes of both its initial and final cells.

One complication that is introduced is that although it is necessary to know $\mathbf{J}$ at the boundary nodes of the grid, there is not strictly a dual cell associated with such nodes, nor consequently a dual cell volume. It is useful to consider a 2D example of this situation before dealing with the 3D problem. Figure 8 shows a 2D grid in the vicinity of a primary node N_P

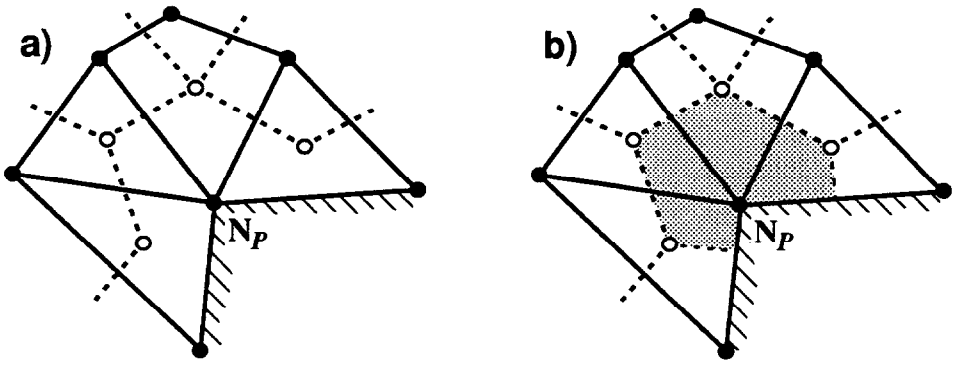

Figure 8. 2D example of a "partial" dual cell at the simulation boundary.

residing on the boundary of the grid. As in Figure 2, primary edges and nodes are depicted as solid lines and filled circles, respectively. Similarly, dual edges and nodes are shown as dashed lines and open circles. As seen in Figure 8a, there is no complete dual cell associated with the boundary node N_P because the node is not completely enclosed by dual edges. However, we can construct new "partial" dual edges by connecting the "hanging" dual nodes to the boundary edges (see Figure 8b). We can then construct a "partial" dual cell (shown as the shaded region in Figure 8b) whose boundary is defined by the original dual edges, the constructed "partial" edges, and the primary edge boundaries.

The procedure for defining a "partial" dual cell in 3D is analogous but more complex. In this case, a given primary node N_P on the boundary has an incomplete set of dual faces (one associated with each of the primary edges connected to N_P *not* in the boundary). If we can augment that set with constructed "partial" dual faces, we can close the volume to the grid boundary, defining a "partial" dual cell whose properties, including the volume, can be determined. It turns out that there will be a new "partial" dual face associated with each of the primary edges connected to N_P that lies in the grid boundary. The path defining this face is chosen to start at the midpoint of this boundary primary edge, connecting to the barycenter of one of the two primary boundary faces that share that primary edge. (The barycenter of a cell

or face is defined to be the mean of the positions of all the nodes that define that cell or face). From there, the path goes to the dual node that is the barycenter of the primary cell bounded by the primary boundary face. From this point, a path of zero or more existing dual edges eventually takes us to a dual node that is the barycenter of the primary cell that is bounded by the other boundary face that shares the original primary edge. The path is then completed by continuing to the barycenter of this other face and back to the starting point at the midpoint of the primary boundary edge. One can visualize this construction by examining Figure 8b, but interpreting N_P to be the midpoint of the boundary primary edge and the solid (dashed) lines to be cross-sections of primary (dual) faces. Note that the constructed face will not in general be planar, and the figure represents a projection of the actual face.

Incorporation of Current Density into the VOLMAX Field Solver

Current density due to the motion of free charge contributes to Ampere's Law. In VOLMAX, Ampere's Law is used to update the normal component of the electric field for each dual face. If the current density is included, the update equation becomes

$$\varepsilon \frac{\partial}{\partial t} \int_S \mathbf{E} \cdot d\mathbf{A} = \oint_C \mathbf{H} \cdot d\mathbf{l} - \int_S \mathbf{J} \cdot d\mathbf{A} \ , \tag{3}$$

where S is the surface of the dual face, and C is the closed contour around the face. Since we know the vector current density ($\mathbf{J}$) at primary nodes, we will approximate its value over a dual face by the mean of its value at the two nodes of the primary edge associated with the dual face, i.e.,

$$\int_S \mathbf{J} \cdot d\mathbf{A} \equiv \left[\frac{\mathbf{J}_1 + \mathbf{J}_2}{2} \cdot \hat{\mathbf{n}}_F \right] A_F \ , \tag{4}$$

where $\hat{\mathbf{n}}_F$ is the dual face normal, $\mathbf{J}_1$ and $\mathbf{J}_2$ are the vector current density at the two primary nodes, and A_F is the area of the dual face.

VOLMAX also updates the vector electric field at the primary nodes using the volume integral form of Ampere's Law, in which case the current density contribution must be taken into account. That update equation then becomes

$$\varepsilon \frac{\partial}{\partial t} \int_V \mathbf{E} dV = \oint_S d\mathbf{A} \times \mathbf{H} - \int_V \mathbf{J} dV \ , \tag{5}$$

where V is the volume of a dual cell and S is the closed surface of that cell (the combined surfaces of all the cell's faces). Since we know current density at the primary nodes from (2), we simply approximate the $\mathbf{J}$ integral term by the product of $\mathbf{J}$ with the dual cell's volume.

Charge Conservation

One of the traditional difficulties associated with PIC simulation is charge conservation. This difficulty arises because the EM field algorithm advances the electric and magnetic fields by integrating Ampere's Law and Faraday's Law (the two Maxwell "curl" equations). Gauss' Law ($\nabla \cdot \mathbf{D} = \rho$) is satisfied implicitly if there is no free charge in the system. However, when there is free charge, Gauss' Law is satisfied only if the continuity equation

( ) is satisfied. Unfortunately, simple choices of particle current density allocation schemes and their associated weight factors do not in general satisfy the continuity equation and a spurious electrostatic field associated with this error in charge conservation results. If of a sufficiently large magnitude, this error can have an unacceptable affect upon the fidelity of the simulation.

Several approaches[21,22] have been taken to correct this problem; we have chosen to implement the technique[22] proposed by Marder, which he refers to as the "pseudo-current" method, but has become commonly known as the "Marder" correction. In this method, a new field is defined,

$$F(\mathbf{x}, t) = \nabla \cdot \mathbf{D} - \rho \tag{6}$$

which represents the error in charge conservation. Ampere's Law is then modified by adding a multiple of the gradient of F to the right-hand side:

$$\partial \mathbf{E}/\partial t = c^2 \nabla \times \mathbf{B} - \mathbf{J}/\varepsilon + d\nabla F,$$

where d is a numerical constant. The added term is referred to as a "pseudo-current" (actually ε times the "pseudo-current") since it appears in the equation in a current-like fashion. It can be shown[22] that F satisfies an inhomogeneous diffusion equation

$$\partial F/\partial t - d\varepsilon \nabla^2 F = -(\partial \rho/\partial t + \nabla \cdot \mathbf{J}).$$

Note that if F is set to zero on the boundaries, the code will diffuse the errors to the boundary. The constant d controls the diffusion rate; its maximum value is limited numerically by the Courant stability constraint $2d\varepsilon\ \Delta t/\Delta x^2 < 1$, where Δx is the 1D-equivalent cell size. For the unstructured grid, we typically choose Δx to be the length of the shortest primary edge.

The implementation of the Marder correction is straightforward on a rectilinear grid. Since electric field components are located spatially along the orthogonal cell edges, the divergence is naturally centered at the cell nodes, which is also were the charge density is located; consequently, F is naturally located at the cell nodes. Gradients of values located at cell nodes are naturally centered on the cell edge connecting two nodes. This is precisely the location of the current density; consequently, the "pseudo-current" ($d\varepsilon\ \nabla F$) can be easily combined with the actual particle current in Ampere's Law.

On an unstructured VOLMAX grid, computation of F is also straightforward. If we consider the volume integral of F over the dual cell that contains a given primary node,

$$\int_V F dv = \int_V \nabla \cdot \mathbf{D}\ dv - \int_V \rho dv = \varepsilon \cdot \oint_S \mathbf{E} \cdot d\mathbf{A} - Q_C, \tag{7}$$

where Q_C is the total charge in the dual cell, S is the surface enclosing the dual cell (the union of all the dual cell's faces), $d\mathbf{A}$ is a vector differential area element of S in the outward normal direction, and we have made use of the divergence theorem to obtain the surface integral term. If we now divide by the volume of the dual cell, we obtain an average value for F at the primary node. VOLMAX already computes the electric field component normal to all dual faces, which together with the dual face areas, allow computation of the surface integral in (7). Since the weighting factors described in previous sections relate particles to primary nodes, we will in fact accumulate total charge at the primary nodes, which can be interpreted as the total charge in the dual cell containing the primary node.

The computation of the gradient of F is less straightforward. The VOLMAX algorithm for advancing Ampere's Law (3) requires the component of current density normal to each dual face. Consequently, we need to be able to compute $\nabla F \cdot \hat{\mathbf{n}}_F$ on the dual face, where $\hat{\mathbf{n}}_F$ is the dual face normal. Since F is located at primary nodes, it is natural to compute the component of the gradient along primary edges ($\nabla F \cdot \hat{\mathbf{u}}_E$, where $\hat{\mathbf{u}}_E$ is a unit vector from node 1 to node 2 along the primary edge). In the rectilinear case, this unit vector is aligned with the associated dual face normal, which is why the gradient of F is aligned with the current density and is straightforward to compute. In contrast, on an unstructured grid, $\hat{\mathbf{n}}_F$ and $\hat{\mathbf{u}}_E$ are in general not aligned, and consequently there are several different approaches to compute $\nabla F \cdot \hat{\mathbf{n}}_F$. All involve determining a vector approximation for ∇F at the primary nodes. There are two different approaches that can be used. First, one can divide its integral over the volume of the associated dual cell by the dual cell's volume. This integral can be converted to a surface integral using the gradient form of the divergence theorem, giving

$$\nabla F \cong [\int_V \nabla F dV]/V_D = [\oint_S \tilde{F} d\mathbf{A}]/V_D \ , \tag{8}$$

where V_D is the volume of the dual cell and $\tilde{F}$ is the approximate value of F at each dual face, which we take to be the mean of the values of F at the two nodes of the primary edge associated with the dual face.

A second approach to approximating ∇F at the primary nodes starts by computing $\nabla F \cdot \hat{\mathbf{u}}_E$ for all primary edges:

$$G_E \equiv \nabla F \cdot \hat{\mathbf{u}}_E \cong (F_2 - F_1)/L_E \ , \tag{9}$$

where F_1 and F_2 are the values of F at the nodes 1 and 2 of the primary edge, respectively, and L_E is the length of the primary edge. Then for a given primary node, ∇F is approximated by a least-squares fit to the G_E for all primary edges containing the given primary node. VOLMAX presently uses this same technique to approximate the vector electric field at primary nodes from normal components of the electric field on the associated dual faces.

Once the primary node vector gradients have been obtained using one of the two methods just described, there are two different approaches to finding the normal component on the dual faces. The first simply approximates the value at the face by the mean of the values at the two nodes of the primary edge associated with the dual face and then takes the dot-product with the face normal, i.e., $\nabla F \cdot \hat{\mathbf{n}}_F$, where

$$\tilde{\nabla F} \cong \frac{1}{2}(\nabla F_1 + \nabla F_2) \ , \tag{10}$$

and ∇F_1 and ∇F_2 are the values of ∇F at the two nodes of the primary edge. The second method is based on the premise that the edge components of the gradient are more accurate than the vector approximations at the primary nodes and that the dual face normal is predominantly in the direction of the associated primary edge. If we decompose the gradient into two components, one parallel to and one perpendicular to the primary edge, and use the vector gradient only to determine the perpendicular component, we obtain the following approximation for the face-normal component of the "pseudo-current":

$$\nabla F \cdot \hat{\mathbf{n}}_F \cong \{[\tilde{\nabla F} - (\tilde{\nabla F} \cdot \hat{\mathbf{u}}_E)\hat{\mathbf{u}}_E] + G_E \hat{\mathbf{u}}_E\} \cdot \hat{\mathbf{n}}_F \ . \tag{11}$$

Particle Emission

There are two primary means of introducing particles into a PIC simulation: via beam emission and via space-charge-limited (SCL) field emission. In either case, particles are introduced from a conducting surface. Clearly, introducing charge anywhere else in the space will violate charge continuity since we would in essence be creating charge out of "thin air". For beam emission, the particles are injected into the simulation with prescribed density and velocity distribution. Consequently, the injection of beam particles is straightforward; at an appropriate rate, we introduce particles at the emission surface, with the correct velocities and with sufficient charge, to produce the prescribed density and velocity distribution.

In contrast, the density and velocity of particles introduced due to SCL emission are not prescribed but is instead determined by the electric fields local to the site of emission. Specifically, an SCL emission surface is assumed to be an unlimited source of free charge and the current extracted from such a surface is limited only by space charge — eventually the electric field that provides the accelerating force for particles leaving the surface is driven to zero by the electrostatic field of the emitted particles' own charge. Consequently, an algorithm for SCL attempts to introduce just enough charge at the emission surface so that the electric field normal to the conducting surface will vanish.

A typical approach to accomplish this is to apply Gauss' Law over the volume of the "partial" dual cell associated with a boundary primary node on an SCL emission surface, i.e.,

$$Q_N = \varepsilon \int_V \nabla \cdot \mathbf{E} dV = \varepsilon \oint_S \mathbf{E} \cdot d\mathbf{A}, \tag{12}$$

where Q_N is the charge at primary node N_P, V is the volume of the "partial" dual cell, and S is its enclosing surface. Remember that the charge accumulated at the primary node N_P represents the total charge in the partial dual cell. The geometry associated with (12) is shown schematically in Figure 9, which shows an emission site at a primary node N_P as well as the dual faces and their corresponding face-normal electric fields. In addition to the "full" dual faces, the enclosing surface includes the partial dual faces as well as the portion of the boundary primary faces intercepted by the partial faces. For SCL, the normal component of electric field at the boundary vanishes and hence the contributions from the boundary primary faces to the surface integral in (12) is zero. To first order, the integrals over the partial dual faces also vanish since the *total* electric along the boundary primary edges that pierce the partial faces is zero. Using this information, assuming that the normal component of the electric field is constant over a dual face, and splitting Q_N into two portions (Q_{add} and Q_{init}), (12) becomes

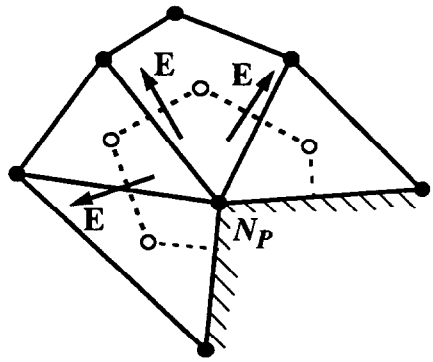

Figure 9. Illustration of use of Gauss' Law for SCL emission algorithm.

$$Q_{add} \cong \varepsilon \sum_F (\mathbf{E} \cdot \hat{\mathbf{n}}_F) A_F - Q_{init}, \tag{13}$$

where the sum is over all the dual faces, $(\mathbf{E} \cdot \hat{\mathbf{n}}_F)$ is the known face-normal electric field, A_F is the area a dual face, Q_{init} is the charge already in the dual cell before emission, and Q_{add} is

the amount of charge that needs to be added to the dual cell to force the surface electric field to vanish (SCL). Typically, the emission from an SCL surface is only one species, or at least multiple species of the same polarity (e.g., only electrons or only positive ions). Consequently, if the sign of Q_{add} is the same as that of the emission species, Q_{add} is the amount of charge to be emitted by the algorithm. If on the other hand, the signs are opposite, there is already too much charge in the cell and no particles will be emitted.

Particle Data Management

In the QUICKSILVER code, for reasons of efficiency, particles are not handled individually, but in groups of many particles. In addition, only a few pieces of information about a single particle need to be persistent (i.e., saved from one timestep to the next); however, much more information is required per particle while they are being processed to determine their motion and allocate that back to the grid. For this reason, particles are in fact arranged in a hierarchical fashion. First particles are grouped into *vectors*; a vector contains the maximum number of particles that will be processed at any given time. The size of a vector is determined by the opposing constraints of having a large vector for efficiency, but a small vector to reduce memory usage required for each particle while they are being processed. These particle vectors are then grouped into *caches*, which contain only persistent, timestep-to-timestep particle data. QUICKSILVER provides extensive methods for efficiently managing particles in this data structure, and it was decided to utilize that structure as much a possible to handle particles moving on the unstructured grid.

First consider the persistent information required to describe the state of a particle — charge, species, spatial position, and momentum. For convenience, we also carry a random number, generated at a particle's time of creation, for each particle. This is useful for several models and diagnostics. Finally, we must save enough information to locate the particle relative to the grid through which it is moving. For a structured multi-block QUICKSILVER grid, this requires four integers, three coordinate indices and a block number, which are typically packed into a single integer. For an unstructured grid, only a single integer, the primary cell containing the particle, is required. Note that this last item is the only difference between the requirements for the two grid types and it is consequently straightforward to use a standard QUICKSILVER cache for particles in the unstructured grid — we simply interpret one member of the cache data structure in a different way. We need to add a new list containing all the caches currently in use for unstructured-grid particles, which is analogous to the one already used for caches for structured-grid particles.

Armed with the cache structure, we can now outline the procedure for combining all the particle-handling components described in this section in order to process particles on the unstructured grid. For each cache of particles on the unstructured grid's cache list:

1. We extract the persistent data of particles from the cache one vector at a time.
2. The particles in the vector are then processed. This involves:
   a. Advancing their momentum and position in time,
   b. Allocating current density and charge to the grid,
   c. Checking if the particle has crossed a boundary requiring some further pro-

cessing (e.g., removal from system),

d. Extracting any requested diagnostic information.

3. The persistent data of any of the vector's particles that remain, after advancement, on the unstructured grid are stored into a cache of already-processed particles for storage until the next timestep.
4. Any particles newly created in the unstructured grid this timestep must be placed in the cache of already-processed particles.

Note that the management approach outlined so far does not account for treatment of particles that, during their processing during a single timestep, move from the unstructured grid to the structured grid, or *vice versa*. Discussion of the management of such particles will be deferred until a later section that describes particle handling through the interface region in detail.

Time Synchronization of Structured-Grid/Unstructured-Grid Solution with Particles

In order to understand the relationship and interaction between the solutions on the two grid types, it is useful to understand the flow in time of the algorithm that updates the fields and particles. First, the algorithm used by QUICKSILVER will be described to show the interaction between fields and particles. Next, the flow of the VOLMAX algorithm will be shown to provide an understanding of the way the solutions in the structured and unstructured grids are combined. Finally, we will describe the algorithm that results as a combination of the two capabilities. A detailed discussion of grid interface issues is deferred to the next section.

In the following descriptions, we will use superscripts to indicate the temporal location of a given value. For example, $\mathbf{E}^k$ represents the electric field at the kth timestep ($t = k\Delta t$). Similarly, $\mathbf{H}^{k+1/2}$ represents the magnetic field at $t = (k + 1/2)\Delta t$. The algorithms are "leap-frogged" in time, i.e., various state variables are located at different times, separated by one-half the timestep, in order to center the numerical approximation to the time derivatives. Variables can be thought of as full-step or half-step values (e.g., $\mathbf{E}^k$ and $\mathbf{H}^{k+1/2}$, respectively) based upon their temporal location.

Original QUICKSILVER Algorithm

The original QUICKSILVER algorithm assumes that at the beginning of the $(k+1)$th timestep [the timestep that advances the simulation from $k\Delta t$ to $(k+1)\Delta t$], the following state variables of the system are known: $\mathbf{E}^k, \mathbf{H}^{k+1/2}, \rho^k, \mathbf{x}^k, \mathbf{p}^{k-1/2}$, where $\mathbf{E}$ and $\mathbf{H}$ are the electric and magnetic fields, ρ and $\mathbf{J}$ are the charge and current densities due to particle motion, and $\mathbf{x}$ and $\mathbf{p}$ are the position and momentum of each particle. q_P is used to denote the charge of a single PIC particle. The algorithm proceeds as follows:

1. Create new particles $(\rho^k \rightarrow \mathbf{x}^k, \mathbf{p}^{k-1/2}, q_P)$
2. Advance all particles
   a. Advance particle momenta $(\mathbf{p}^{k-1/2}, \mathbf{E}^k, \mathbf{H}^{k+1/2} \rightarrow \mathbf{p}^{k+1/2})$
   b. Advance particle position $(\mathbf{x}^k, \mathbf{p}^{k+1/2} \rightarrow \rho^{k+1}, \mathbf{J}^{k+1/2}, \mathbf{x}^{k+1})$
3. Advance the electric field $(\mathbf{E}^k, \mathbf{H}^{k+1/2}, \mathbf{J}^{k+1/2} \rightarrow \mathbf{E}^{k+1})$
4. Advance the magnetic field $(\mathbf{H}^{k+1/2}, \mathbf{E}^{k+1} \rightarrow \mathbf{H}^{k+3/2})$.

Note that in this simple flow description some steps have been omitted, for example, the required normalization of the current and charge densities as well as the application of boundary conditions to the electric and magnetic fields. Also note that QUICKSILVER actually advances the magnetic flux ($\mathbf{B}$) rather than the magnetic field intensity ($\mathbf{H}$). The flux and field intensity are related by the permeability of the medium (μ), i.e., $\mathbf{B} = \mu\mathbf{H}$. Finally, since QUICKSILVER's current density allocation scheme conserves charge exactly, it does not include a Marder correction. However, if it did, we would need to add a new step after step 3 that

computes F^{k+1} from $\mathbf{E}^{k+1}$ and ρ^{k+1}, and step 3 would need to include a "pseudo-current" computed from the gradient of F to advance the electric field.

Original VOLMAX Algorithm

The VOLMAX algorithm does not have the complication of particles and their associated space charge, but it does have complications due to performing field advancement on two separate grids, each with its own timestep. At the beginning of the $(k+1)$th timestep, the following state variables of the system are known: $\mathbf{E}^k, \mathbf{H}^{k+1/2}, \mathbf{E}_0^k, \mathbf{H}_{1/2}^k$, where we have added the notation that variables with subscripts are unstructured-grid variables. The subscript denotes after which sub-step of the unstructured-grid solution the variable refers. For example, ξ_j^k refers to the value of the variable ξ at $t = (k + j/N_U)\Delta t$, where N_U is the number of sub-timesteps used by the unstructured-grid solver in a single timestep of the structured-grid solver. Note that this definition implies that $\xi_{N_U+j}^k = \xi_j^{k+1}$. The VOLMAX algorithm proceeds as follows:

1. Advance electric field on structured grid $(\mathbf{E}^k, \mathbf{H}^{k+1/2} \rightarrow \mathbf{E}^{k+1})$
2. Advance unstructured-grid fields over N_U sub-timesteps
   Loop over j from 1 to N_U
   a. Advance unstructured-grid electric field $(\mathbf{E}_{j-1}^k, \mathbf{H}_{j-1/2}^k \rightarrow \mathbf{E}_j^k)$
   b. Advance unstructured-grid magnetic field $(\mathbf{H}_{j-1/2}^k, \mathbf{E}_j^k \rightarrow \mathbf{H}_{j+1/2}^k)$
   State at end-of-loop: $\mathbf{E}_{N_U}^k, \mathbf{H}_{N_U+1/2}^k \equiv \mathbf{E}_0^{k+1}, \mathbf{H}_{1/2}^{k+1}$
3. Advance magnetic field on structured grid $(\mathbf{H}^{k+1/2}, \mathbf{E}^{k+1} \rightarrow \mathbf{H}^{k+3/2})$.

Figure 10 shows a timeline over one timestep for the special case that N_U equals two. The field quantities referred to in the above algorithm are shown in their proper location along the timeline. From this diagram it is easy to see that $\mathbf{E}_2^k \equiv \mathbf{E}_{N_U}^k = \mathbf{E}_0^{k+1}$

For this case, a brief word about boundary conditions is in order. After step 2a above, the electric fields on primary edges lying in the wrapper outer boundary are not correct, but need to be in order to proceed to step 2b. These values need to be supplied by the structured-grid solution at the same spatial locations. Note, however, that the temporal location of the two solutions does not match — the latest structured-grid values are ahead (in time) of the

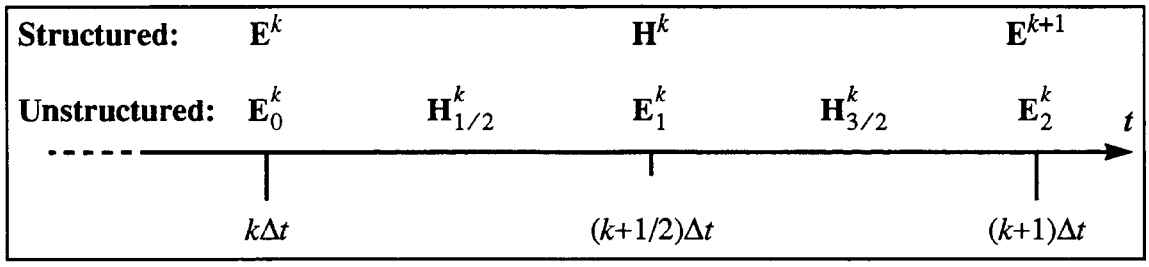

Figure 10. Timeline diagram showing the temporal location of the field quantities for both the structured and unstructured grids ($N_U = 2$).

unstructured-grid solutions until the last sub-timestep. Consequently, the proper time location is obtained by interpolation from the latest structured-grid values as well as values from previous timesteps. For example, if we use linear interpolation, we obtain

$$E_j^k \equiv C_0 E^k + C_1 E^{k+1},\ C_0 = 1-\delta,\ C_1 = \delta,\ \delta = j/N_U \quad , \tag{14}$$

where E is an edge component of the electric field in the wrapper outer boundary. Note that for purely EM applications, VOLMAX is typically used with quadratic interpolation, i.e.,

$$E_j^k \equiv C_{-1} E^{k-1} + C_0 E^k + C_1 E^{k+1}, \tag{15}$$

where

$$C_{-1} = \frac{1}{2}\delta(\delta-1),\ C_0 = 1-\delta^2,\ C_1 = \frac{1}{2}\delta(\delta+1).$$

Similarly, after step 1 above, the electric fields on primary edges lying in the wrapper inner boundary are not correct, but need to be in order to proceed to step 3. However, at the end of step 2, we know the values of the field at these same spatial locations and at the same time. Consequently, at the end of step 2, these values are supplied to the structured grid directly from the corresponding values in the unstructured grid.

It should be noted that when N_U is one, the time advancement algorithms on both grids are completely aligned and the need for interpolation is gone. Indeed, in this case both (14) and (15) degenerate to the latest structured-grid electric field edge component, E^{k+1}. Also note that in this case, all electric fields are advanced (steps 1 and 2a), then all magnetic fields are advanced (steps 2b and 3).

The Combined QUICKSILVER–VOLMAX Algorithm

The combined QUICKSILVER–VOLMAX algorithm assumes that at the beginning of the (k+1)th timestep, the following structured-grid state variables of the system are known:

$$\mathbf{E}^k, \mathbf{H}^{k+1/2}, \rho^k, \mathbf{x}^k, \mathbf{p}^{k-1/2}.$$

In addition, the following unstructured-grid state variables are also assumed to be known:

$$\mathbf{E}_0^k, \mathbf{H}_{1/2}^k, Q_0^k, F_0^k, \mathbf{x}_0^k, \mathbf{p}_{-1/2}^k,$$

where Q is the total charge accumulated at a primary node and F is the measure of charge conservation [cf. (6)] at a primary node. The algorithm proceeds as follows:

1. Create new particles on structured grid $\quad (\rho^k \rightarrow \mathbf{x}^k, \mathbf{p}^{k-1/2}, q_P)$
2. Advance particles on structured grid
   a. Advance particle momenta $\quad (\mathbf{p}^{k-1/2}, \mathbf{E}^k, \mathbf{H}^{k+1/2} \rightarrow \mathbf{p}^{k+1/2})$
   b. Advance particle position $\quad (\mathbf{x}^k, \mathbf{p}^{k+1/2} \rightarrow \rho^{k+1}, \mathbf{J}^{k+1/2}, \mathbf{x}^{k+1})$
3. Advance the electric field on structured grid $\quad (\mathbf{E}^k, \mathbf{H}^{k+1/2}, \mathbf{J}^{k+1/2} \rightarrow \mathbf{E}^{k+1})$
4. Advance unstructured grid fields over N_U sub-timesteps
   Loop over j from 1 to N_U

a. Create new particles on unstructured grid $(Q^k_{j-1} \rightarrow \mathbf{x}^k_{j-1}, \mathbf{p}^k_{j-3/2}, q_P)$

b. Introduce any particles that have come from the structured grid (at step 2 above) for sub-timestep j

c. Advance particles on unstructured grid

momenta $(\mathbf{p}^k_{j-3/2}, \mathbf{E}^k_{j-1}, \mathbf{H}^k_{j-1/2} \rightarrow \mathbf{p}^k_{j-1/2})$

position $(\mathbf{x}^k_{j-1}, \mathbf{p}^k_{j-1/2} \rightarrow Q^k_j, \mathbf{J}^k_{j-1/2}, \mathbf{x}^k_j)$

d. Advance unstructured-grid electric field $(\mathbf{E}^k_{j-1}, \mathbf{H}^k_{j-1/2}, F^k_{j-1}, \mathbf{J}^k_{j-1/2} \rightarrow \mathbf{E}^k_j)$

e. Compute F_j $(\mathbf{E}^k_j, Q^k_j \rightarrow F^k_j)$

f. Advance unstructured-grid magnetic field $(\mathbf{H}^k_{j-1/2}, \mathbf{E}^k_j \rightarrow \mathbf{H}^k_{j+1/2})$

State at end-of-loop: $\mathbf{E}^k_{N_U}, \mathbf{H}^k_{N_U+1/2}, Q^k_{N_U}, \mathbf{x}^k_{N_U}, \mathbf{p}^k_{N_U-1/2}, F^k_{N_U}$
$\equiv \mathbf{E}^{k+1}_0, \mathbf{H}^{k+1}_{1/2}, Q^{k+1}_0, \mathbf{x}^{k+1}_0, \mathbf{p}^{k+1}_{-1/2}, F^{k+1}_0$

5. Advance magnetic field on structured grid $(\mathbf{H}^{k+1/2}, \mathbf{E}^{k+1} \rightarrow \mathbf{H}^{k+3/2})$.

As in the original VOLMAX algorithm, after step 4d above, the electric fields on primary edges lying in the wrapper outer boundary are not correct, but need to be in order to proceed to step 4e. However, if these field are corrected using just the original method outlined in the previous section, there will be a slight error. This is because the computation of the structured-grid electric field on the wrapper outer boundary in step 3 is in error to the extent that it did not include the effects of particles moving over that timestep on the unstructured grid in the wrapper cell layer. These effects are not even known until step 4c is completed for the last sub-timestep. To correct the error, we must accumulate a correction term due to this current over all the sub-timesteps, and add it to the time-interpolated value from the structured grid. Note that because of the aforementioned error in the structured-grid electric field on the wrapper outer boundary (and all the edges connecting the wrapper inner and outer boundaries), when we supply field values from the unstructured grid to the structured grid for the wrapper inner boundary at the end of step 4, we must also supply the corrected values for these additional structured-grid electric fields.

Charge will be accumulated at the nodes on both the wrapper inner and outer boundaries on both grids in steps 2 and 4c. Consequently, charge on both grids at these nodes must be combined before any calculation requiring charge, on either grid, is performed. For example, to compute F on the unstructured grid for any of these nodes requires that the charge that was accumulated on the corresponding structured-grid nodes be included. Since the charge is co-located in time with the electric field, its structured-grid value similarly needs to be time-interpolated to the current unstructured-grid time value [cf. (14) or (15)]. Also, after completion of step 4e, the values of F on the outer wrapper are not correct since not all of the electric field values required for their computation are within the domain of the unstructured grid. They will be needed to compute the "pseudo-current" term on the next timestep. Consequently, F for these boundary nodes will need to be computed on the corresponding nodes and subsequently supplied to the unstructured grid.

Structured-Grid/Unstructured-Grid Mesh Interface Issues

The hybrid grid algorithm embodied in VOLMAX basically integrates Maxwell's equations over one timestep independently in both the structured and unstructured regions of the simulation domain. As described in the introduction, these two independent solutions are then coupled at the boundary that separates the two grids. The coupling is accomplished by allowing a one-cell thick overlap, the wrapper layer (see Figure 2), between the two grid regions; field values one cell inside the structured grid provide the boundary condition for the unstructured-grid solution and conversely field values one cell inside the unstructured grid provide the boundary condition for the structured-grid solution.

When free charge is added (i.e., PIC is used) the situation becomes more complicated. However, the view that the solution in each grid region is driven at the boundary by the other grid's solution is still basically valid. The complications arise from the need to pass more field information back and forth between the two grids (**H**, **J**, ρ, F) and that **J** and ρ values of the two solutions in the shared wrapper layer must be combined to account for the fact that particles moving on both grids can contribute to those quantities. This also means that the electric field advancement on the structured grid, which happens before the corresponding unstructured-grid advancement, cannot correctly compute the electric field on the edges of the wrapper cells, since we don't yet know the current provided by particles moving in the wrapper cell on the unstructured grid; consequently, we need to correct those fields *a posteriori*. A final complication is that we need to properly transition particles from one grid to the other as they move through the wrapper cell region. These issues will be treated in detail in the remainder of this section.

Additional Requirements for Field Quantities at the Grid Interface

In a discussion of the requirements for properly treating field quantities at the structured/unstructured grid interface, we will continually be referring to values defined on both primary edges and nodes in the vicinity of the interface. Consequently, we will define some terms to locate those edges and nodes that will be used throughout the remainder of this section. First, all primary edges located in the wrapper outer boundary (cf. Figure 2) will be referred to as *outer edges*. All primary edges located in the wrapper inner boundary will be referred to as *inner edges*. Similarly, all nodes lying in the wrapper outer (or inner) boundary will be referred to as *outer* (or *inner*) *nodes*. Finally, all primary edges that connect an inner and outer node will be referred to as *connecting edges*. Note that outer edges connect two outer nodes and that inner edges connect two inner nodes. These three types of edges are the set of all edges for which the two grids share values of electric field, and are also the edges where structured-grid current densities are accumulated that are initially missing contributions of particles moving on the unstructured grid. The two types of nodes (inner and outer) are the set of all nodes for which the two grids share values of charge density (charge on the unstructured grid) as well as the nodes at which the unstructured grid accumulates current density contributions that need to be included *a posteriori* in the structured-grid solution. These are also the nodes at which the two grid solutions must interact to properly compute the scalar

measure of charge conservation F.

From the previous discussion of the QUICKSILVER-VOLMAX algorithm on page 30, we remember that the structured-grid solution for the electric field steps ahead of the unstructured grid-solution, which then catches up with one or more sub-timesteps. Structured-grid values for electric fields on the outer and connecting edges are interpolated in time at each sub-timestep to bound the unstructured-grid solution using (14) or (15). Using the same approach, charge density, which is temporally collocated with electric field, can be interpolated from the structured grid to the current sub-timestep of the unstructured-grid solution. Note that the situation for current density, which also needs to be shared between the two grids is somewhat different. The structured-grid solution is located at the half-timestep, meaning it is advanced in time one-half timestep ahead of the unstructured-grid solution. Note that this means that, for cases with multiple sub-timesteps ($N_U > 1$), the unstructured-grid solution will actually compute currents that advance ahead of the time of the last structured-grid current density. Consequently, the algorithm for passing structured-grid current density information to the unstructured grid will actually involve extrapolation. Since extrapolation (other that constant) can often be unstable, we have chosen to simply use the last structured-grid value at $(k+1/2)\Delta t$ for all unstructured-grid sub-timesteps from $k\Delta t$ to $(k+1)\Delta t$. Note that this constant extrapolation is equivalent to viewing the structured-grid current density as a piecewise-constant function of time, centered at the half-timestep.

As stated earlier, the original VOLMAX algorithm interpolated the structured-grid electric fields at outer and connecting edges to supply boundary conditions to each sub-timestep of the unstructured-grid solution. After all sub-timesteps are complete, the values of the unstructured-grid electric field at inner connecting edges are supplied as a boundary condition to the structured-grid solution. (It should be noted that, although not strictly required for bounding the unstructured-grid solution, the connecting edge fields will be computed to have the same value on both grids due to the topology of the wrapper and the design of the unstructured-grid solver.) However, when particles are added, the structured-grid field values on the outer and connecting edges are not correct due to motion of particles moving in the wrapper layer on the unstructured grid. However, on a sub-timestep by sub-timestep basis, we can compute a correction to the electric field at any one of these edges, i.e.,

$$\tilde{E}_j^k = E_j^k + \Delta E_j,$$

where $\tilde{E}_j^k$ is the corrected edge electric field at the jth sub-timestep, and E_j^k is the electric field interpolated from the structured grid using (14) or (15). The correction after j sub-timesteps, ΔE_j, is given by

$$\Delta E_j = -\frac{\Delta t}{\varepsilon N_U} \sum_{i=1}^{j} (\mathbf{J}_1 + \mathbf{J}_2)_{i-1/2}^k \cdot \hat{\mathbf{u}}_E \,, \tag{16}$$

where $\Delta t/N_U$ is the unstructured-grid sub-timestep, $\mathbf{J}_1$ and $\mathbf{J}_2$ are the currents collected on the edge's two nodes from particles moving in the wrapper cell, and $\hat{\mathbf{u}}_E$ is the edge-directed unit vector. This correction is applied to the corresponding unstructured-grid edge electric fields after step 4d of the QUICKSILVER-VOLMAX algorithm. At the end of the last sub-timestep, when the VOLMAX algorithm would normally supply unstructured-grid field values at inner connecting edges back to the structured grid, we also now need to supply the unstructured-

grid field values at outer and connecting edges, since they include the correction term (16) and are now correct.

Note that for connecting edges, one of the two nodes referenced in (16) will be an inner node, which accumulates current both for particles moving in the wrapper layer as well as particles moving in unstructured-grid cells interior to the wrapper inner boundary. Since we want to include only the current of particles moving in the wrapper layer in this correction, we need to separate the current collected at inner nodes into that due to wrapper cell particles and that due to non-wrapper cell particles. On the other hand, the vector current density needed by the field algorithm [see (3)–(5)] at inner nodes needs to include both the wrapper and non-wrapper contributions. It should be pointed out that implementation of the capability to separate current and charge contributions at the inner nodes in this fashion requires significant additions to the VOLMAX's database.

Strictly speaking, since **J** on the structured grid has already been used to advance the structured-grid electric fields at the time (16) is computed, it is not necessary to correct the structured-grid currents themselves but to instead correct the affected electric fields, as described previously. However, for diagnostic purposes, it may be desirable to correct the current density on these structured-grid edges anyway. If the number of unstructured-grid sub-timesteps per structured-grid timestep (N_U) is odd, the unstructured-grid current density at sub-timestep (N_U+1)/2 is collocated in time with the structured-grid current density, and consequently that single sub-timestep's current density contribution to (16) can be added to the corresponding structured-grid current. If N_U is even, the mean of the contributions from substeps $N_U/2$ and ($N_U/2$)+1 is collocated with the structured-grid current density, and that mean can be added to the corresponding structured-grid current.

The correction described by (16) allows the structured-grid electric fields affected by particle motion on the unstructured grid to be properly determined. Similarly, we need to insure that electric fields computed on the unstructured grid that are affected by particle motion on the structured grid are treated properly. This translates to making sure that the vector current density on the unstructured grid on the inner nodes includes the current of particles moving in the wrapper layer of the structured grid. Since current density on the structured grid is spatially located on the cell edges (collocated with the electric field), currents that affect the inner nodes lie on the inner and connecting edges. For example, Figure 11 shows an x-y cross-section of the interface between the structured and unstructured grids. Note that an implied third subscript (n) has been omitted from the currents shown in the figure to simplify the notation. The unstructured-grid node N_{Inner} maps to the structured-grid location (l,m,n). Particles moving in wrapper cells on the structured grid (shaded region) contribute to the current densities bounding the wrapper cells but not those beyond the wrapper inner boundary; e.g., for the example in Figure 11, $J_{Xl,m}$ on the structured grid never accumulates any current; the other three J's shown (as well as the J_Z components not shown) do accumulate current. Note that a reasonable

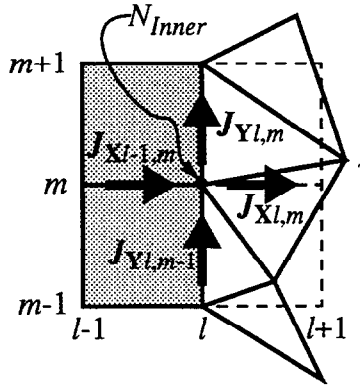

Figure 11. Diagram showing structured-grid current density contributions to unstructured-grid inner nodes.

approximation to each vector component of the structured-grid current density at N_{Inner} is obtained by taking the mean of the two edges aligned with that component, i.e.,

$$\tilde{J}_X(l,m,n) = \frac{1}{2}[J_X(l-1,m,n) + J_X(l,m,n)],$$

$$\tilde{J}_Y(l,m,n) = \frac{1}{2}[J_Y(l,m-1,n) + J_Y(l,m,n)],$$

$$\tilde{J}_Z(l,m,n) = \frac{1}{2}[J_Z(l,m,n-1) + J_Z(l,m,n)], \qquad (17)$$

where we are relying that any contributions from structured-grid edges beyond the wrapper inner boundary are zero. Thus we obtain for the total vector current density at the inner node N_{Inner} at the jth sub-step of the kth timestep,

$$(\mathbf{J}_{Total})_j^k = (\mathbf{J}_{Unstructured})_j^k + \tilde{\mathbf{J}}^k,$$

where $\mathbf{J}_{Unstructured}$ is the vector current density allocated to the node by particles moving on the unstructured grid only, and $\tilde{\mathbf{J}}^k$ is the structured-grid correction term whose components are given by (17). Note that since the structured-grid current density is temporally located at the half-timestep, we will assume the structured-grid contribution to be constant for all sub-timesteps.

As discussed previously in the section on computation of particle weights (see page 15), the magnetic fields on the unstructured grid must be interpolated to the primary nodes from their location at the dual nodes in order to use those weights to determine the magnetic field at the location of any particle. This interpolation can be accomplished in a straightforward manner by taking an average of the magnetic fields at the nodes of the dual cell containing the primary node. However, this technique cannot be used for outer (wrapper) nodes since an enclosing dual cell does not exist. In fact, this information can only be provided using the field values from the corresponding nodes of the structured grid. Since the magnetic field is temporally located at the half-timestep, we will, in the same spirit as our treatment of current density, assume that the structured-grid magnetic field at time $(k+1/2)\Delta t$ can be assumed constant over the entire unstructured-grid advancement from $k\Delta t$ to $(k+1)\Delta t$. Consequently, it is sufficient to set the unstructured-grid magnetic field at the outer primary nodes from the corresponding values on the structured grid before proceeding to step 4 in the hybrid grid algorithm outlined on page 30.

A similar analysis of the proper treatment of charge in the vicinity of the interface is needed. This is simpler than the analysis for current density for two reasons: first, structured-grid charge is not needed in the time advancement algorithm until after any corrections due to particles on the unstructured grid can be made; and second, the charge on both grids is spatially located at the nodes. On the other hand, charge density is stored on the structured grid and charge is stored on the unstructured grid, requiring a conversion from one grid to the other. Also, since charge density on the structured grid is temporally located on the full-timestep, we will need to interpolate using (14) or (15) to the sub-timestep of the unstructured-grid solution.

The overlap in the charge allocation for the two grids occurs on the inner and outer nodes. Analogous to the electric field, the structured-grid charge density is advanced in time a full timestep, and then the unstructured-grid charge "catches up" over one or more sub-timesteps. Consequently, if computations are made using inner or outer structured-grid charge density values *before* that density has been corrected for charge collected on the corresponding unstructured-grid nodes, those computations must be corrected as well. For example, if we have computed the charge error measure F for outer nodes on the structured grid (which uses the structured-grid charge density at those nodes), we will need to correct that value of F to reflect the additional charge collected at the nodes on the unstructured grid after the unstructured-grid advancement is completed. Inspection of the hybrid algorithm described on page 30 reveals that we have deferred the use of structured-grid values of charge density in subsequent calculations until the unstructured-grid advancement (step 4 of the algorithm) is complete in order to avoid this complication. Consequently, we only need to ensure that the time-interpolated structured-grid charge density is properly included in computations requiring the charge in the unstructured-grid advancement and that after that advancement, the charge density associated with the charge at unstructured-grid inner and outer nodes is added to the charge density at the corresponding nodes on the structured grid. Note that both these steps require the conversion of charge to charge density (and vise versa) by the division (multiplication) of an appropriate volume. This volume is in all cases the volume of the equivalent dual cell on the *structured* grid.

It is worth describing in more detail the computation of the charge error measure F for outer nodes. This is needed on the unstructured grid as a boundary condition. Since QUICKSILVER's algorithm for current allocation conserves charge exactly (within numerical roundoff), F is zero (or essentially zero) at all structured-grid nodes that do not overlap with the unstructured grid. Consequently, we do not need to include "pseudo-current" terms for any of the edges connected to these nodes; thus we need to include "pseudo-current" only on the unstructured grid. We also know that any error in charge conservation (nonzero F) at the outer nodes is due only to charge and current density allocated on the unstructured grid. Thus if we provide the unstructured grid with the value of F computed from the structured-grid electric field and charge density at an outer node, we can correct it at each sub-timestep for the charge collected from particles on the unstructured grid and for any electric field corrections (16) on the edges connected to the outer node. In QUICKSILVER's structured-grid algorithm, F can be computed at any node (l,m,n) on the structured grid by finite difference approximation:

$$\begin{aligned} F(l,m,n) = \varepsilon[&D^{x,m}_{lmn}E_x(l-1,m,n) + D^{x,p}_{lmn}E_x(l,m,n) + D^{y,m}_{lmn}E_y(l,m-1,n) \\ &+ D^{y,p}_{lmn}E_y(l,m,n) + D^{z,m}_{lmn}E_z(l,m,n-1) + D^{z,p}_{lmn}E_z(l,m,n)] \\ &- \rho(l,m,n) \end{aligned} \qquad (18)$$

where the various D terms are difference coefficients, each associated with one of the edge electric fields that contributes to the divergence.

Note that if (l,m,n) corresponds to an outer node of the wrapper N, some of the electric field edges in (16) are affected by particles in the wrapper layer of the unstructured grid. Consequently, we can compute F at each unstructured-grid sub-timestep as

$$F_j^k(N) = F^k(l,m,n) + \varepsilon \sum_e D_{lmn}^{p,q} \Delta E_j(e) - Q_j^k(N)/V(l,m,n), \qquad (19)$$

where $F^k(l,m,n)$ is computed from the structured-grid fields at time $k\Delta t$ using (18), $Q(N)$ is the charge collected at the unstructured-grid node N at the jth sub-timestep, and $V(l,m,n)$ is the volume of the structured-grid dual cell. The sum in (19) is over all outer and connecting edges that are connected to node N, $D^{p,q}$ is the appropriate difference coefficient for each edge from (18), and the ΔE_j is the edge electric field correction given by (16).

Although QUICKSILVER's structured-grid algorithm does not presently use F to compute a "pseudo-current" correction, there are reasons why it might be desirable to do so. For example, one might want to switch to a less-noisy, but non-conserving algorithm, in which case a "pseudo-current" correction would be appropriate. In this case, it should be noted that only minor modifications would be required to accommodate the change. The hybrid grid algorithm described on page 30 would need to add a new step, immediately following step 3, that would compute F^{k+1}, using the algorithm described in (18). In (19), $F^k(l,m,n)$ would need to be replaced by a time-interpolated value [cf. (14) or (15)] of the structured-grid F.

Particle Handling Through the Interface Region

As particles move, they interact with the fields defined on the grid in two ways: they use the fields on the grid to determine the EM forces that control their motion and they provide current that affects the fields on the grid. Consequently, since the wrapper layer provides a one-cell thick layer of cells in both the structured and unstructured grids, particles moving in that layer can interact with either grid. Note, however, that once such a particle moves across the wrapper inner boundary, it can no longer interact with the structured grid; similarly, once that particle crosses the wrapper outer boundary it can no longer interact with the unstructured grid. Since a particle starting in given cell can always potentially leave that cell in a single timestep, it is impossible, with only a single-cell wrapper layer to guarantee that over one timestep the interaction of a particle starting in the wrapper cell in either grid can be treated properly on that grid throughout the entire timestep. For example, a particle starting in a wrapper cell of the structured grid can cross the wrapper inner boundary. To correctly account for its interaction with the grid, at least that portion of its path inside the inner boundary must be accounted for on the unstructured grid. Consequently, we have a choice: either we increase the thickness of the wrapper layer to two cells, or we are forced to allow for allocating at least a portion of a particle's motion over a single timestep to a different grid than the particle started in. Note, on the other hand, that a particle initially located in an interior (non-wrapper) cell of the unstructured grid cannot possibly leave the unstructured grid (including the wrapper) in a single *structured*-grid timestep (remember that the structured-grid timestep can be an integer multiple of the unstructured-grid timestep).

Since the VOLMAX algorithm advances the structured-grid fields a single timestep *before* it advances the unstructured-grid fields with a series for one or more sub-steps, the structured-grid fields repeatedly leap ahead of the unstructured grid and the unstructured-grid fields are then "caught up" during the unstructured portion of the advancement. As a result, particles on the structured grid will be advanced before particles on the unstructured grid. Consequently, it

is exceedingly difficult to allocate the motion of an unstructured-grid particle that crosses the wrapper outer boundary back to the structured grid because the structured-grid current densities have in fact already been used to advance the structured-grid electric field. In contrast, it is less difficult to allocate the motion of a structured-grid particle that crosses the wrapper inner boundary back to the unstructured grid because we haven't yet even advanced the unstructured-grid particles for this timestep.

Based upon the arguments made above, we have chosen to adopt the following model to deal with this issue:

- We will continue to use a one-cell thick wrapper layer —the added complexity in adding an additional layer of wrapper cells in both VOLMAX and PREVOL is not worth the benefit gained.
- The motion of any particle on the structured grid that crosses the wrapper inner boundary will be divided into two segments at the point where it crosses the inner boundary. Only the motion along the portion of the path within the wrapper layer is allocated to current density on the structured grid; treatment of the remainder of the path will be deferred for processing on the unstructured grid.
- Any particle on the unstructured grid that is located in a wrapper cell after completion of all sub-steps of the current timestep will be moved to the corresponding structured-grid wrapper cell for further processing there on the next timestep. Consequently, no particle will ever *start* a timestep in an unstructured-grid wrapper cell, and therefore cannot leave the unstructured grid in the course of the timestep.

The structured-to-unstructured transition is shown schematically in 2D in Figure 12. Note that the need for this transition is detected by determining that the particle's final position is in a structured-grid cell that is "beyond" the wrapper layer (indicated by the dashed cells in the figure). This requires that all such structured-grid cells be tagged with this property. In addition, we also tag such cells with the number of an unstructured-grid wrapper cell that corresponds to the adjacent structured-grid wrapper cell. This information allows us to efficiently find the unstructured-grid cell into which the particle passes at the wrapper inner boundary using the particle location algorithm outlined in the previous section.

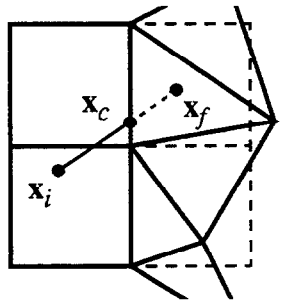

Figure 12. Illustration of treatment of a particle leaving the structured grid.

Once it has been determined that the particle moved beyond the wrapper inner boundary, the particle's motion from $\mathbf{x}_i$ to $\mathbf{x}_c$ (see Figure 12) is allocated to the appropriate structured-grid current densities. We then need to save the particle's persistent state information (charge, momentum, position, random number, species, grid location) for later use during the completion of that particle's advancement on the unstructured grid. Note that the position saved is the point $\mathbf{x}_c$ where the particle enters the unstructured grid and the grid location is the number of the unstructured-grid cell into which the particle immediately passes at the inner boundary. However, we also need to save one extra piece of information — the fraction of the timestep for which the particle still needs to be advanced. We assume this to be the fraction of the path from $\mathbf{x}_i$ to $\mathbf{x}_f$ that is beyond the wrapper inner boundary ($|\mathbf{x}_f - \mathbf{x}_c| / |\mathbf{x}_f - \mathbf{x}_i|$). This

extra information is needed so that the unstructured-grid particle handler can determine in which sub-timestep to introduce the particle and what fraction of that first sub-step to advance the particle. In this first sub-step, the particle is advanced as any other particle, except that we do not use the fields to advance the momentum, and the position is advanced only by the supplied fraction of the sub-timestep. Note that if there remains only one sub-step (or fraction thereof), this algorithm will move the particle to precisely the location originally computed by the structured-grid advancement, $\mathbf{x}_f$. Otherwise, the final position after the last sub-step will be somewhat different due to the refinement of the timestep on the unstructured grid.

Some discussion of the caching strategies for the persistent data required for particles moving from the structured to unstructured grids is in order. Since the timestep fraction is also required for each of these particles, they do not readily fit in the existing cache structure. In addition, there are typically relatively few of these particles in comparison to the standard particle cache size. Finally, since only a subset of such particles need to be processed in a given sub-timestep of the unstructured-grid solver, it is convenient to put particles that will re-enter the simulation on different sub-timesteps in different caches in order to avoid processing each of them every sub-timestep. For these reasons, a separate structure of smaller caches, each with space for one extra word per particle, was set up. In addition, a separate list of caches is maintained for each sub-step. Then, at each sub-step of the unstructured-grid solver, the particle handler processes only those caches in the appropriate list, thus avoiding needless processing.

Although the treatment of particles moving from the unstructured grid to the structured grid is much simpler, a few details are worth mentioning. When, at the end of the last sub-timestep of the unstructured-grid solver a particle is determined to be located in a wrapper cell, instead of storing its persistent data to a cache of already-processed unstructured-grid particles, we instead add that data to an appropriate cache of already-processed *structured-grid* particles. However, as discussed in an earlier section, the required persistent data is somewhat different; a particle's grid location is determined by a block number and three coordinate indices rather than by a primary cell number. Consequently, we need to be able to map the unstructured-grid wrapper cell in which the particle is located to its structured counterpart. This is accomplished as follows. Every cell on the unstructured grid has a tag in the CellInfo array that indicates whether or not it is a wrapper cell. This tag is used to determine whether or not a particle is located within a wrapper cell. For cells that are wrapper cells, the CellInfo array also contains a pointer to the cell's position on a list of all wrapper cells. With this pointer we can access the block number and coordinate indices that map the cell to the corresponding wrapper cell on the structured grid. These mapping values are packed into a single integer for each element of the list of wrapper cells. Note that the indirection to an intermediate list of wrapper cells and the packing of the values describing the structured-grid location are both used for purposes of efficiency in memory usage.

Testing

The testing of the software developed in this project falls into two broad categories: algorithm testing and integrated testing. First, the implementations of each of the algorithms developed over the course of the project were tested in a stand-alone environment. For example, the particle location algorithm described on page 13 was tested by randomly choosing a cell (N) within an unstructured grid and a random position within or slightly beyond the range of the grid ($\mathbf{x}_f$). Starting from the barycenter of N ($\mathbf{x}_i$), we attempt to locate the cell containing $\mathbf{x}_f$, or in the case that $\mathbf{x}_f$ is beyond the grid, the face and spatial location where the line connecting $\mathbf{x}_i$ and $\mathbf{x}_f$ leaves the grid. Note that this test is in general more severe than is required since particles can only move a limited distance in a single advancement, whereas this test allows for a search that could extend across the entire grid. Each algorithm developed, as well as the routines that initialized the extensive new data structure required by the new algorithms, were similarly tested.

Although the verification of each of the new algorithms through this stand-alone testing provides an efficient approach to making this new code work, it is not sufficient, and must be augmented by integrated testing of the entire code. This is the only way to insure that all of the individual algorithms work together to provide the correct answer to a real problem.

Perhaps the most basic integrated test that can be performed consists of a closed rectangular box. Within the box is a structured grid containing an embedded unstructured grid over a smaller sub-volume inside the box. A 2D cross-section of this geometry is shown in Figure 13. A low-current, high-energy beam of electrons is emitted from one face of the box. By low-current, high-energy beam we mean that its current is sufficiently low relative to its energy that its space charge does not significantly affect its momentum. Consequently, we expect all beam particles emitted at the same time to arrive at the box's far face at precisely the same time, with the same momentum, and at their original transverse (to beam direction) spatial position. This should be true of all particles, whether or not they cross through the unstructured grid region of the simulation. This tests the handling of particles in the unstructured portion of the grid as well as their transition between the two grid regions. Specifically, we test the following issues:

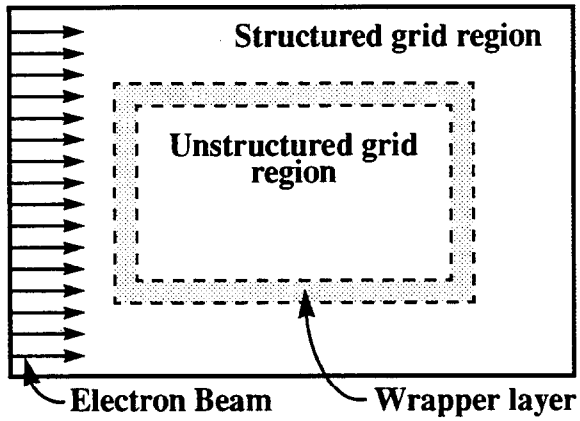

Figure 13. 2D view of simple "beam-in-a-box" test simulation.

- that particles are properly placed in the special cache for structured-to-unstructured transitions,
- that particles are properly extracted from those caches at the proper sub-timestep and with the correct fractional sub-timestep,
- that the unstructured-grid particle advancement works properly,
- that the new cache structure for unstructured-grid particles works properly,

- When re-entering the wrapper layer from the unstructured grid, that particles are properly placed back into a cache of structured-grid particles,
- that no particles become lost or otherwise "confused" throughout this entire process.

Although this list of features tested may seem a limited subset of the features that need to be tested, it should be realized that it does exercise a very large fraction of the new code written, and in fact, some of the most complex, particularly with regard to particle cache management and the interaction of those caches. In fact, this test, before it was successfully completed, unearthed the majority of bugs found in the entire testing process.

A logical extension of the previous test is to place a conductor inside the unstructured grid. Note that such an inclusion actually eliminates a portion of the unstructured mesh and the volume of the conductor is "outside" the grid. For example, by inserting a conducting plate whose cross-section intercepted some fraction of the beam traversing the unstructured portion of grid, we were able to test if particles encountering the grid boundary were properly eliminated from the simulation and that the cache management associated with that operation was performed correctly. Several variations of this test were successfully performed.

The next simulation that we will describe tests most aspects of the interaction of particle motion with the EM fields. The geometry for this test problem is shown in Figure 14. It consists of a capacitor formed by two concentric cubic conductors whose sides are 44 cm and 6 cm. A high-energy beam is injected from a subsurface of one of the faces in such a way that the entire beam is intercepted by the inner cube. The beam current is a 2.5 ns pulse with amplitude of one ampere, which will charge the inner cube with 2.5×10^{-9} coulombs. This charge will cause a potential difference between the two cubes of a few hundred volts (based upon a simple estimate of the capacitance of this geometry). This geometry is simulated in two ways: first, as a control, we use only a structured mesh, and second, we embed the inner cube in an unstructured grid which is then embedded in an interior sub-region of a structured mesh. The structured mesh is uniform with a cell size of two cm. The unstructured grid is bounded by a cubic outer wrapper boundary, concentric with the two conductors and whose side is 24 cm. By comparing time histories of voltages at several locations between these two simulations, we can verify the proper operation of several code features. Specifically, this comparison tested the following additional features:

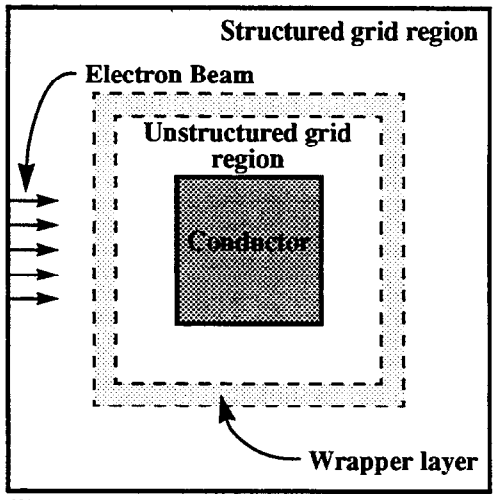

Figure 14. 2D view of simple "beam-charged capacitor" test simulation.

- that the fields driven by particle motion on the unstructured grid are consistent with that motion,
- that the issues associated with allocating current and charge in the wrapper layer on both grids are properly handled,
- that we properly treat the now larger set of field information that must be exchanged at the interface between the grids.

The unstructured-grid version of this simulation was performed in three distinct ways. Two of

the simulations were done with a purely rectilinear unstructured grid that exactly matched the structured grid, one with N_U (the number of unstructured-grid sub-steps per structured-grid timestep) equal to one and the other with N_U equal to three. The third simulation used a tetrahedral grid with ~9200 primary cells; the minimum primary cell edge was slightly less than one cm and N_U was eight.

First we will describe the comparison of the two hybrid-grid simulations with the embedded rectilinear unstructured grid to the purely structured-grid simulation (which we assume to be correct since the QUICKSILVER code is extremely well validated for problems of this type). Standard macroscopic diagnostics such as voltage, current, field energy, as well as particle count, charge, and energy differ by less than 0.1 percent between the three simulations. This is not particularly surprising since the grids are in fact the same — the only real difference between the structured-grid solution and the two hybrid solutions is that the algorithm for allocating charge is different for the unstructured grid region of the hybrid solutions. However, the result is significant since all of the complications of sub-timestepping have been tested. In addition, the hybrid-grid simulations do not conserve charge exactly, and the results give us our first indications that the degree of charge conservation on the unstructured grid is acceptable. In fact, for this test the "pseudo-current was not used to reduce the error in charge conservation.

The more interesting comparison is of the hybrid-tetrahedral-grid simulation to the original purely structured-grid simulation. A comparison of the voltage measured through the axis of the electron beam between the inner and outer cubical conductors is shown in Figure 15. The figure shows the voltage rising until ~4 ns, by which time the entire beam has been collected on the inner conductor. The voltage continues to oscillate about a constant d.c. value due to interaction of the current pulse with the resonances of the structure. Note that the agreement between the simulations is quite good with differences less than 5%. In fact, since the tetrahedral mesh has finer spatial resolution (as much as a factor of two) and eight times the temporal resolution, one would expect a somewhat different answer. For this simulation, the "pseudo-current" correction for charge conservation was turned off; consequently, the unstructured-grid simulation could be somewhat in error due to that effect. Figures 16 and 17 show similar comparisons of voltage. Figure 16 shows the analogous measurement to that of Figure 15, but on the opposite side of the structure. Figure 17 shows a similar measurement between the inner and outer faces in the direction transverse to the beam (because of symmetry, all four such measurements should be and are the same). Note that both exhibit similar behavior to the voltage measured on the beam axis and the differences between the two solutions are comparable.

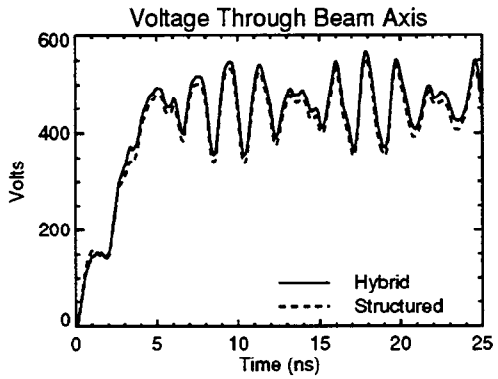

Figure 15. A comparison of voltage through the beam axis between a structured-grid and hybrid-tetrahedral-grid simulation.

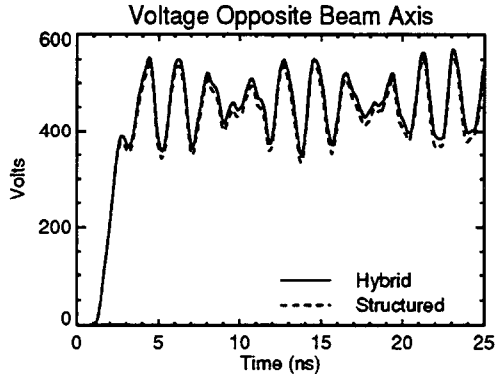

Figure 16. A comparison of voltage at far end of capacitor between a structured-grid and hybrid-tetrahedral-grid simulation.

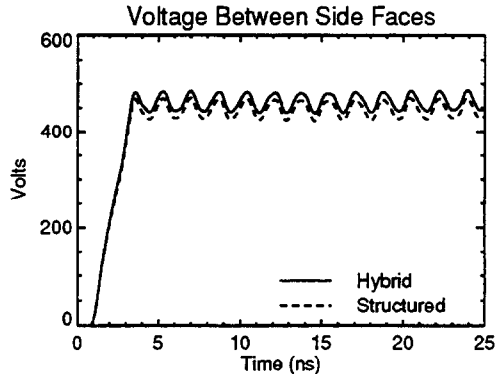

Figure 17. A comparison of voltage transverse to the beam axis between a structured-grid and hybrid-tetrahedral-grid simulation.

The final test to be reported here, we take the geometry of Figure 14, with the same hybrid-tetrahedral grid, but reverse the direction of the beam such that the electrons flow from a face of the *inner* cube to the corresponding face of the outer cube. As a result of this change, we expect the d.c. voltage to reverse, and oscillations should be similar in character. However, they should be somewhat different due to the fact that during the electron transit time at the beginning and end of the pulse, the spatial distribution of current density will be different due to the propagation direction of the beam. Note that this simulation tests the feature of beam emission on the unstructured grid, which up to this point has not been tested. Figure 18 shows a comparison of the voltages measured along the axis of the beam (analogous to Figure 15 for the previous test). Note that we have reversed the sign of the voltage for purposes of display. Again, the agreement is comparable to the previous test and the voltage behaves as expected. This comparison is representative of comparisons of the other macroscopic observables in the simulations. For reference purposes, the same measurement from the original beam orientation (from Figure 15) is superimposed.

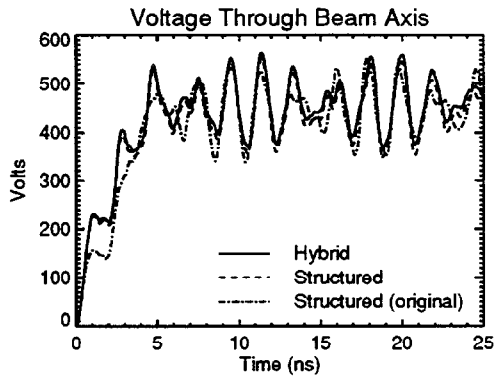

Figure 18. A comparison of voltage through the beam axis between a structured-grid and hybrid-tetrahedral-grid simulation for the reversed beam case.

Early Estimates of Numerical Performance

Using timings from our various test simulations, we are able to make some estimates of the computational efficiency of the new particle-handling features of the code. As a point of reference, we observe that the cost of advancing the fields one sub-timestep for one cell of the unstructured grid is approximately a factor of 3–6 more expensive (depending upon the type of unstructured-grid cell) than a similar one-timestep advancement of the fields for one cell on a rectilinear structured grid. A similar comparison of the time required to advance one particle

one sub-timestep on the unstructured grid to the time required to advance one particle one timestep on the structured grid, shows the unstructured-grid advancement to be approximately four times as expensive. Note that the times stated for unstructured-grid field and particle advancement are for a single *sub-timestep*; consequently, the time ratios quoted must be multiplied by a factor of N_U if we want to know the cost ratio between the two grid types for the same time advancement increment. In general, we are quite pleased with the efficiency observed for the unstructured-grid particle advancement. since we had in fact originally anticipated that it might be as high as a factor of eight. Also note that no particular attempt has yet been made to optimize the unstructured-grid particle advancement so there are opportunities to improve its performance somewhat more.

Current Status and Future Development

At the end of the project supporting this work, most of the basic algorithms required to apply the PIC method to hybrid grids have been designed, implemented, and integrated into a new QUICKSILVER/VOLMAX simulation code. Most, but not all, of the features of this new code have been successfully tested in an integrated manner for some simple test problems. This section of the report will detail the code's current status by way of describing the basic capabilities that remain to be integrated and subsequently tested. From that point, we will discuss the development of some more advanced capabilities, that although beyond the initial scope of this project, would be useful to extend the range and/or improve the usability of the code.

Basic Capabilities Not Yet Implemented

There are two basic capabilities that have not yet been implemented into the integrated code. The first is SCL emission, the algorithm for which is described in a previous section (page 25). As currently envisioned, this algorithm will emit from user-specified boundary faces, using charge information at each face's nodes to determine the amount of charge to inject into the simulation. Note that most of the components required for such an algorithm already exist — we already compute all the terms that contribute to the calculation of charge to add at each emission node (13) and the beam emission algorithm contains the tools to locate and build lists of the surface faces that comprise an emission surface. The only new code remaining to be implemented is the computation of the nodal charges and their subsequent distribution to the associated emission faces.

The second basic capability that remains unimplemented is the interaction of particles moving on the unstructured grid with symmetry boundaries, specifically those boundaries with mirror (also referred to as perfect-magnetic-conductor or PMC boundaries) and periodic symmetry. Currently, the particle advancement routine contains tentative code, as yet unused, to implement this interaction. We could proceed no farther, because the support for these boundaries in the VOLMAX field solver was not available during the project period. However, the mirror algorithm became available about one month after our project's completion and we expect the periodic algorithm to be ready in the very near future. We now need to adjust our tentative code to reflect the final details of the field algorithm's mirror implementation and test it. When the periodic boundary is available, we will need to repeat this process.

Although presently implemented, we are not currently satisfied with all aspects of the performance of the "pseudo-current" charge conservation algorithm. In our previous discussion (see page 22), we outlined two distinct approaches to compute the vector ∇F at the primary nodes and also two distinct approaches to computing the dual-face-normal components of ∇F. In our current implementation, we use the divergence integral algorithm (8) to compute the primary node vector, and use the node-average technique (10) to obtain the face normal components. After observing the performance, we believe that the alternate approach to computing the face-normal components of the gradient, using (9) and (11), will

provide much better accuracy and finer resolution of its spatial variation. We would also like to assess the alternate "least-squares" technique for computing the primary node vectors from the edge-directed components defined by (9).

Advanced Development for the Future

Most of the development issues that will need to be addressed in the future are related to the ease-of-use for the code as a production simulation tool. These divide into three broad categories:

- Problem definition and setup,
- Integrated diagnostic capability,
- Capability and performance for large and/or complex simulations.

We will discuss each area in turn.

Presently, in order to perform a simulation, we must set up the unstructured portion of the mesh using the I-DEAS CAD package. Concurrently, the structured portion of the grid must also be constructed using QUICKSILVER's preprocessor, MERCURY. The user is entirely responsible for insuring that the overlapping wrapper layers of the two grids are totally consistent (not an easy task for a simulation with any complexity). In addition, it is also the user's responsibility to insure that any geometric structure is also consistent between the two grids. Note that such structure must be separately provided to both I-DEAS and MERCURY by significantly different means. It is highly desirable to remove as much of this burden as possible from the user, and place it upon the tools themselves. This will require significant changes to both MERCURY and our customizations of I-DEAS or else we will need to find or develop a single integrated tool to build the entire hybrid grid as a single entity.

A related issue is that of diagnostics. For example, consider the work that is now required to obtain something as simple as a voltage from a hybrid grid simulation, such as those shown in Figures 15–18. Since the voltage is obtained by integrating the electric field along a path, if that path lies in both grids, we need to include the contribution of each sub-path. Presently, we need to specify the structured-grid portion of the path to MERCURY, which causes QUICKSILVER to compute that quantity and store the accumulated time history in a PFF file. Similarly, we need to also identify the unstructured-grid portion of the path in I-DEAS by tagging each node of the edges that comprise that path with a special tag. For a complex system this can be a laborious and error-prone process. Once these nodes are tagged, VOLMAX can obtain the integral over its sub-path at each timestep. These are then saved, each in its own file and in a format different from the PFF format used for the structured-grid data, for post-processing. After the simulation is complete, the user must, by using the data manipulation capabilities of the post-processing tool, e.g., PFIDL, read in both components of the diagnostics from their respective sources and then combine them, with the proper signs, into the single desired time history. This is just an example of several diagnostics, such as energy and flux integrals, snapshots in time of planes of field data, etc., that are presently quite difficult to obtain.

It should be noted that we have begun to design and develop the required tools to

simplify this procedure. For instance, for the example just described we have outlined a procedure by which the integration path could be specified to MERCURY alone, which would know how to subdivide the path into its structured-grid and unstructured-grid portions. QUICKSILVER could then find the best fit to the unstructured-grid portion of the requested path by constructing a sequence of connected edges through the unstructured grid. Using this information, its routines to gather history information could be generalized to access and include the unstructured-grid portions of the integral with the structured-grid portions, and finally write only one set of data to one PFF file. We have presently written the routines that will allow MERCURY to subdivide the path and to construct the unstructured-grid sub-path, but need to integrate them, along with the other needed modifications, in the code. A similar tactic could be used to integrate most if not all diagnostics for hybrid grid simulations.

As a final point, we will touch on the issue of performing large simulations. Presently, both the QUICKSILVER and VOLMAX codes are being ported to run on Sandia's Intel TeraFlop[23] distributed-memory, massively-parallel supercomputer. We need to include the new features that have been added by this project to support PIC techniques on hybrid grids into the parallel implementations of QUICKSILVER and VOLMAX.

References

1. O. Buneman, *Relativistic Plasmas*, O. Buneman and W. Pardo, eds., Benjamin, New York 1968.
2. M. E. Jones, "Electromagnetic PIC Codes with Body-Fitted Coordinates," *Proc. 12th Conference on the Numerical Simulation of Plasmas*, San Francisco, CA, Sept. 20–23, 1987.
3. T. Westermann, *Nucl. Instrum. Methods A* **263**, 271 (1988); D. Seldner and T. Westermann, *J. Comput. Phys.* **79**, 1 (1988).
4. Scott Brandon, Lawrence Livermore National Laboratory, private communication.
5. John Ambrosiano, Los Alamos National Laboratory, private communication.
6. D. J. Riley and C. D. Turner, *IEEE Microwave and Guided Wave Letters* **5**, 284–286 (1995).
7. D. B. Seidel, M. L. Kiefer, R. S. Coats, T. D. Pointon, J. P. Quintenz, and W. A. Johnson, "The 3-D, Electromagnetic Particle-In-Cell Code, QUICKSILVER." in *The CP90 Europhysics Conference on Computational Physics*, Armin Tenner, Ed., World Scientific, Amsterdam, 1991, pp. 475–482.
8. J. P. Quintenz, D. B. Seidel, M. L. Kiefer, T. D. Pointon, R. S. Coats, S. E. Rosenthal, T. A. Mehlhorn, M. P. Desjarlais, and N. A. Krall, *Laser and Particle Beams* **12**, 283-324 (1994).
9. D.J. Riley and C.D. Turner, *IEEE Antennas Propagat. Mag.* **39**, 20–33 (1997).
10. D.J. Riley and C.D. Turner, *11th Annual Review of Progress in Applied Computational Electromagnetics (ACES) Symposium Digest*, 435–444 (1995).
11. L. P. Mix, R. S. Coats, and D. B. Seidel, "*PFIDL: Procedures for the Analysis and Visualization of Data Arrays*," presented at the 1st Biennial Tri-Laboratory Engineering Conference on Computational Modeling, Pleasanton, California, Oct. 31–Nov. 2, 1995.
12. D. B. Seidel, R. S. Coats, M. L. Kiefer, T. D. Pointon, and L. P. Mix, "*PFF — A Compact, Machine-Independent File Format for Simulation Data*," presented at the 9th Biennial CUBE Symposium, Santa Fe, New Mexico, Nov. 27–30, 1990.
13. K. S. Yee, *IEEE Trans. Antennas Propagat.* **14**, 2155–2163 (1966).
14. O. Buneman, "*Fast Numerical Procedures for Computer Experiments on Relativistic Plasmas*," in Relativistic Plasmas, O. Buneman and W. Pardo, Eds., New York: Benjamin, 1968, pp. 205–219.
15. B. B. Godfrey, Time-Biased Field Solver for Electromagnetic PIC Codes," presented at the 9th Conference on Numerical Simulation of Plasmas, Evanston, Illinois, June 30–July 2, 1980.
16. G. Mur, *IEEE Trans. Electromagnetic Compatibility* **23**, 1191–1196 (1982).
17. D. B. Seidel and J. P. Quintenz, "Diodes and Magnetic Insulation," in *Computer Applications in Plasma Science and Engineering*, Adam Drobot, Ed., Springer-Verlag, New York, 1991, p. 47.
18. Z. Bi, K. Wu, C. Wu, and J. Litva, *IEEE Trans. Microwave Theory Tech.* **40**, 774–777 (1992).
19. J.-P. Berenger, *J. Comp. Physics* **114**, 185–200 (1994).
20. D. J. Riley and C. D. Turner, "The VOLMAX Transient Electromagnetic Modeling System, Including Sub-Cell Slots and Wires on Random Non-Orthogonal Cells," to appear in *Proc. 1998 Applied Computational Electromagnetic Society (ACES) Symposium*, Monterey, CA, March, 1998.
21. C. K. Birdsall and A. B. Langdon, *Plasma Physics via Computer Simulation*, McGraw-Hill, New York 1985, p. 359.
22. B. Marder, *J. Comput. Phys.* **68**, 48 (1987).
23. T. G. Mattson, D. Scott, S. Wheat, "A TeraFLOP Supercomputer in 1996: The ASCI TFLOP System," *Proc. International Conference on Parallel Processing*, Honolulu, HI, April, 1996, pp. 84–93.

DISTRIBUTION:

3 Lawrence Livermore National Laboratory
Attn: Scott Brandon, L–018
Neil Madsen, L–154
David Steich, L–154
P.O. Box 808
Livermore, CA 94550

1 Los Alamos National Laboratory
Attn: John Ambrosiano, F633
P. O. Box 1663
Los Alamos, NM 87545

1 University of Illinois at Urbana-Champaign
Attn: Prof. A. C. Cangellaris
Dept. of Electrical and Computer Engineering
1406 W. Green St.
Urbana, IL 61301-2991

1 Centre D'Etudes de Gramat
Attn: Rene Vezinet
Departement Electromagnetisme et Rayonnements Ionisants
46500 Gramat
FRANCE

1 MS 0188 LDRD Office, 4523
1 MS 1165 J. Polito, 9300
1 MS 1166 J. D. Kotulski, 9352
5 MS 1166 D. J. Riley, 9352
1 MS 1166 G. J. Scrivner, 9352
5 MS 1166 C. D. Turner, 9352
1 MS 1186 M. P. Desjarlais, 9533
1 MS 1186 T. A. Mehlhorn, 9533
1 MS 1186 T. D. Pointon, 9533
1 MS 1186 S. A. Slutz, 9533
1 MS 1186 R. A. Vesey, 9533
1 MS 1186 R. S. Coats, 9542
5 MS 1186 M. L. Kiefer, 9542
1 MS 1186 R. W. Lemke, 9542
1 MS 1186 L. P. Mix, 9542
5 MS 1186 M. F. Pasik, 9542
20 MS 1186 D. B. Seidel, 9542
1 MS 1186 S. E. Rosenthal, 9573
1 MS 1187 B. M. Marder, 9571
1 MS 1188 C. L. Olson, 9541
1 MS 1188 R. A. Hamil, 9512
1 MS 1188 J. S. Wagner, 9512
1 MS 1190 D. L. Cook
1 MS 1193 J. E. Maenchen, 9531
1 MS 1194 D. H. McDaniel, 9573
1 MS 1194 R. B. Spielman, 9573
1 MS 1195 J. P. Quintenz, 9502
1 MS 9018 Central Technical Files, 8940-2
5 MS 0899 Technical Library, 4916
2 MS 0619 Review & Approval Desk, 12690 For DOE/OSTI